Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

Filipe Manuel Dias Ferreira

Caracterização de Redes Sem Fios Multihop Não Planeadas

Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

Filipe Manuel Dias Ferreira

Caracterização de Redes Sem Fios Multihop Não Planeadas

Tese de dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Dr. Rui Luís Andrade Aguiar, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho à minha família, amigos e namorada.

o júri

presidente

Prof. Dr. José Luís Guimarães Oliveira
professor associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Paulo Alexandre Ferreira Simões
professor auxiliar do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Prof. Dr. Rui Luís Andrade Aguiar
professor associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

"If you think you can do a thing or think you can't do a thing, you're right."
por Henry Ford

Nada se consegue sem trabalho, nada se consegue sem esforço, tudo depende da dedicação e da vontade com que se pretende atingir um objectivo. Conseguir ou não fazer algo, está certo, só depende da própria pessoa para tornar uma delas a realidade.

Todo o trabalho por mim desenvolvido ao longo deste tempo dependeu do apoio incondicional de várias pessoas.

Ao meu pai António, à minha mãe Isaura, à minha irmã Mónica e seu marido Hugo, à minha madrinha Emília, ao meu tio Edgar, à minha tia Guilhermina e aos meus avós, que apesar de não estarem presentes estão sempre no meu coração, aos meus amigos Luís Filipe, Marco Oliveira, Pedro Coelho e Pedro Oliveira, à minha namorada Catarina, aos amigos "*verders"* que fiz durante este percurso universitário e ao João, meu padrinho do curso, e a todos aqueles que em vários momentos me apoiaram, um agradecimento em especial, pois com estes puder contar o seu apoio incondicional e sua confiança em qualquer momento.

Gostaria de agradecer também ao orientador desta tese Prof. Doutor Rui Luís Andrade Aguiar, e à colaboração dos professores Diogo Nuno Pereira Gomes e João Paulo Silva Barraca por todo o tempo e esforço dedicado nas suas valiosas contribuições.

A minha grande consideração e estima por todos vocês e um obrigado por me ajudarem a conseguir tornar isto, uma realidade!

Filipe Dias

palavras-chave

Redes sem fios, IEEE 802.11, Interferências nas ondas rádio, Warbiking, Redes sem fios em malha, Simuladores de redes.

resumo

A presente dissertação propõe-se realizar o mapeamento dos pontos de acesso 802.11 existentes da cidade de Aveiro. Recorrendo ao GPS obteve-se a localização de cada ponto.

Com este mapeamento foi possível introduzir estes dados num ambiente de simulação para, posteriormente, com os dados obtidos na simulação, proceder-se a uma análise em detalhe dos dados obtidos e assim caracterizar a topologia da rede mapeada. Finalmente com todos os dados recolhidos, foram tiradas conclusões gerais sobre o estudo.

Para este estudo contribuiu, um trabalho também realizado sobre uma experiência prática onde a densidade de terminais num meio envolvente a uma dada rede sem fios é comparado com a largura de banda disponível nas redes sem fios que operam no mesmo meio em diversas situações.

Para realizar todo este trabalho foram desenvolvidos *scripts* em Python e C++.

keywords

Wireless Networks, IEEE 802.11, Radio Interferences, Warbiking, Wireless Mesh Networks, Networks Simulators.

abstract

This dissertation tries to map the 802.11 access points in Aveiro city. Using a GPS device, every access point location in the city was gathered.
Using this mapping information, it was possible, using a simulated environment, to come up with detailed data regarding the mapped network toppology. Finally, using all the obtained information, some general conclusions about the study were made.
Another experiment also contributed to this study, regarding a network with dense distribution of terminals, in which the available bandwidth is comparable with the one on the wireless networks operating in a variety of conditions.
To enable this study, several Python and C++ scripts/applications were developed.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

AIEE	American Institute of Electrical Engineers
AODV	Ad-Hoc On-Demand Distance Vector
AP	Access Point
APSK	Amplitude Phase Shift Keying
BPSK	Binary Phase Shift Keying
BS	Base Station
BSHC	Base Station Hybrid Coordinator
BSS	Basic Service Set
CCK	Complementary Code Keying
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DBPSK	Differential Binary Phase Shift Keying
DC	Direct Current
DCF	Distributed Coordination Function
DDR2	Double Data Rate 2
DQPSK	Differential Quadrature Phase Shift Keying
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
ESS	Extended Service Set
EUA	Estados Unidos da Améroca
FCC	Federal Communication Commission
FDM	Frequency Division Multiplexing
GPS	Global Positioning System
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Electronics Engineers
IPU	Iwate Prefectural University
IRE	Institute of Radio Engineers
ISM	Industrial, Scientific and Medical
LTS	Long Term Support
MAC	Media Access Control
MB	MegaByte
Mbits/s	Megabits per second
MDA	Mesh Deterministic Access
Mesh STA	Mesh Station
MIMO	Multiple-Input Multiple-Output
MP	Mesh Point
Muni Wi-Fi	Municipal Wi-Fi
NDIS driver	Network Driver Interface Specification
NICT	National Institute of Information and Communications Technology
NS – X	Network Simulator - [2,3]
OFDM	Orthogonal Frequency Domain Multiplexing
OLSR	Optimized Link State Routing

OSI	Open Systems Interconnection
PA	Ponto de Acesso
PCS	Physical Carrier Sense
QAM	Quadrature Amplitude Modulation
QOS	Quality of Service
QSTA	QOS Supporting 802.16e Station
RAM	Random Access Memory
RTS	Request To Send
S.O.	Sistema Operativo
SDM	Spacial Division Multiplexing
SDRAM	Sincronous Dynamic Random Access Memory
SNR	Signal to Noise Ratio
SRX	Super Range Xpress
SS	Subscriber Stations
SSID	Service Set Identity
STA	Station
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USI	Universal Scientific Industrial
WECA	Wireless Ethernet Compatibility Alliance
WEP	Wired Equivalent Privacy
WI-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WIP	Work In Progress
WLAN	Wireless Local Area Network
WLAN – SWG	Wireless Local Area Networks Standard Working Group
WMN	Wireless Mesh Network
WPA	Wifi Protected Access
WPA-PSK	Wifi Protected Access Pre Shared Key

Capítulo 1

Introdução

1.1 Motivação

A inovação tecnológica na área das telecomunicações revolucionou o mundo nos últimos tempos. Há poucos anos equipamentos com *Bluetooth, Global Positioning System (GPS) e Wi-Fi*, eram escassos e de pouca utilização. Hoje são praticamente indispensáveis. Todo o tipo de ligação existente era por fio, o que se tornava incómodo, provocando dependência.

Esta evolução prende-se ao facto das pessoas necessitarem de mobilidade. Estes equipamentos são formas simples que transportam informação em movimento. Graças a todos os equipamentos sem fios, o conforto aliou-se a esta necessidade, tornando o uso destes quase ininterrupto.

A crescente utilização da Internet como fonte de informação fez com que a internet sem fios se tornasse mais comum nas indústrias e, essencialmente, nas casas de habitação. Surge então a necessidade da existência de redes sem fios que acolhessem necessidades sociais. Estas redes são criadas pelos próprios utilizadores, sem nenhum planeamento definido. Desta forma, qualquer pessoa, adquirindo um ponto de acesso, pode passar a ser membro integrante de uma rede com uma topologia *wireless*. Dois exemplos reais são a WiFi Tastic [**1**] e a rede FON, sendo que esta segunda, já com bastante adesão por parte de várias pessoas em múltiplos países, que, adquirindo

pontos de acesso específicos, partilham uma percentagem da sua largura de banda para a infra-estrutura de comunicações que esta alberga [**2**]. Estas redes encontram-se actualmente em expansão. Qualquer pessoa, em qualquer parte do mundo poderá fazer parte de uma rede destas. A topologia implantada por este tipo de rede é denominada assim *Work In Progress (WIP)*, pois cada ponto de acesso desta rede, tem que disponibilizar internet, e para isso, é obrigado a ter um acesso prórprio em cada ponto de acesso.

A internet sem fios é uma tecnologia que usufrui de ondas rádio para transmissão de informação, apresentando assim, vantagens e desvantagens perante a internet por fio. Num caso extremo, podemos considerar que estas redes podem cobrir toda uma cidade, com algumas limitações. Desta forma, é necessário um estudo aprofundado à possível topologia não planeada que se pode apresentar nas cidades, constituída pelos pontos de acesso dos seus habitantes. Para tal, é preciso analisar a cobertura possível numa cidade, a quantidade e densidade de equipamentos, e sua distância média para estudarmos alguns factores de qualidade de serviço possíveis nestas redes.

Várias cidades, são hoje objectos de estudo sobre o tipo e quantidade de redes sem fios que nela operam. Este tipo de estudos tem como objectivo principal qual a tendência tecnológica, e a que ponto esta se torna consistente, segura e eficaz. Questões de segurança e de mau uso dos pontos de acesso, é reflectido nestes estudos. Para além das redes sem fios, hoje, é posto em jogo o uso de serviços móveis 3G lado a lado. No entanto, os custos suportados por uma rede sem fios, é bem menor, tornando-se assim, e com perspectiva de futuro, a melhor solução encontrada, para fornecer serviços a uma cidade [**3**].

Paris, foi em 2006 uma cidade à qual foi escolhida como alvo para um estudo prático de análise de todos os pontos de acesso [**4**]. O primeiro objectivo pretendido neste estudo, era mapear todos os pontos de acesso 802.11 b/g na cidade. Com isto pretendiam, em primeira mão juntar-se a mais algumas cidades na Europa, e divulgar mais informação de carácter tenológico sobre a sua cidade, e em segunda mão, mostrar o cenário de impacto que as redes sem fios apresentam nos dias de hoje, com a sua crescente utilização. Então, neste estudo efectuado, apresentam-se valores de segurança, valores sobre os fabricantes do material usado para as redes sem fios, valores estatísticos sobre o *Service Set Identity (SSID*) e finalmente, valores de densidade dos canais.

Nos Estados Unidos de América, foram localizados geograficamente cerca de 5,5 milhões de pontos de acesso através de *wardriving*. Este estudo, realça as cidades de *Las Vegas, Atlanta, San Francisco, Seattle, Boston* e *Manhattan*. Para cada uma das cidades, o número de pontos de

acesso é relacionado com a correspondente área e aspectos demográficos, tais como população, obtendo a densidade de pontos de acesso por pessoa e por km^2. Com este valor será possível avaliar o desempenho das redes, qual a saturação do espectro e até que ponto, alguns destes pontos de acesso poderão tornar-se públicos [**5**].

Cidades como Copenhaga (Dinamarca), Londres (Reino Unido), Caracas (Venezuela), Varsóvia (Polónia) e Monterrei (México) foram cidades alvos deste tipo de estudo. De um forma geral os objectivos são comuns a todas: caracterizar os pontos de acesso localizados geograficamente e relacionar os mesmos com dados demográficos, para descrever-se o ponto actual da evolução tecnológica na área das redes sem fios.

Na cidade de estudo, Aveiro, não existe registo de estudos deste género. Sabe-se da existência de algumas infra-estruturas conhecidas, nomeadamente a *eduroam* na Universidade de Aveiro, mas da qual oferecem serviços apenas aos alunos universitários, não abrangendo assim os habitantes da cidade. A *wi-fi* da Portugal Telecom é também outra infra-estrutura que se encontra disponível apenas em locais públicos, tais como Centro Comercial Glicínias e Fórum de Aveiro. Contando com estas e outras mais, para além das instalações privadas, efectuou-se uma análise primitiva a todos os pontos localizados. Assim partindo deste princípio será possível obter uma caracterização da cidade de Aveiro em relação à topologia de infra-estruturas 802.11 encontradas.

Os objectivos são idênticos, e parte dos resultados que serão avaliados, são também os mesmos, nomeadamente a densidade de canais. No entanto, o estudo pretendido, é apenas um passo para a permitir a construção de uma rede sem fios em malha espalhada pela cidade, utilizando a topologia não planeada encontrada.

Este tipo, é utilizado para responder as necessidades que os seus habitantes necessitam. Existem cidades que optaram de forma planeada, construírem este tipo de redes, outras optam por conceitos que se apresentam úteis, o caso da FON e WiFi Tastic, já expostos anterioemente. Analisando esta situação, é objectivo futuro, colocar algo assim à disposição em Aveiro.

De forma sucinta, é notável, que este estudo, apresentará novos dados para a cidade de Aveiro, bem como trouxe para as outras cidades. Sabendo da existência de vários tipos de redes espalhadas pelo mundo com as mesmas necessidades, partindo-se do zero, será possível obter dados e relacioná-los para concluir o estudo realizado na cidade de Aveiro.

1.2 Objectivos

Esta dissertação, proposta pelo Instituto de Telecomunicações de Aveiro [**6**], tem como objectivo geral o mapeamento de todos os pontos de acesso 802.11 da cidade de Aveiro, e o estudo dos serviços que poderão ser fornecidos numa rede sem fios global que se instalasse na cidade. Assim, a nível prático a tese, apresenta-se dividida em três partes distintas: mapeamento, simulação e caracterização. Contudo, para melhor compreensão, torna-se também necessário, como complemento, um estudo global a todos os processos envolventes, nomeadamente, os problemas com os canais rádio na transmissão.

Para o mapeamento dos pontos de acesso na cidade é utilizado um sistema GPS, de forma a obter uma localização fidedigna dos pontos de acesso. Com a primeira fase concretizada, a simulação é o passo seguinte. Esta segunda fase serve para que num ambiente simulado se obtenha uma caracterização das medidas de qualidade de serviço numa eventual rede global. Esta fase, é necessária para que posteriores conclusões sobre os serviços que esta rede, hipoteticamente formada como uma comunidade, poderia prestar aos seus utilizadores.

Finalmente, em jeito conclusivo e crítico, pretende-se avaliar todas as características obtidas anteriormente, e expor a viabilização ou não de uma rede sem fios espalhada pela cidade.

1.3 Estrutura da dissertação

Esta dissertação de mestrado é composta por cinco capítulos que descrevem vários assuntos correlacionados entre, com o presente capítulo introdutório já incluído, onde é explicado todo o objectivo desta dissertação. No final, o quinto capítulo, é um capítulo dedicado às conclusões do presente trabalho. A estrutura desta dissertação, apresenta-se então resumidamente da seguinte forma:

Capítulo 1: descrição do contexto do trabalho e os principais objectivos propostos.

Capítulo 2: apresentação introdutória explicativa e descritiva das redes sem fios, nomeadamente, do standard *Institute of Electrical and Electronics Engineers* (IEEE) 802.11. São descritos vários aspectos que caracterizam esta norma, bem com o

seu funcionamento. Por fim, são apresentados casos reais em aplicações específicas dos vários modos de funcionamento do standard IEEE 802.11.

Capítulo 3: apresentação de um estudo complementar sobre efeitos da densidade de terminais. Neste capítulo são apresentadas as influências práticas do acesso a uma rede exposta a um meio saturado. É apresentada toda a metodologia utilizada, os resultados e conclusões.

Capítulo 4: no capítulo fulcral desta tese é feita uma caracterização de uma rede sem fios não planeada na cidade de Aveiro; neste contexto é apresentado também todo o estudo efectuado que corresponde à parte prática da dissertação, fazendo assim, uma descrição de toda a metodologia, desde a aquisição dos dados ao seu tratamento. É ainda exposto e descrito todo o ambiente de simulação usado para o estudo.

Capítulo 5: no último capítulo desta dissertação são sumarizadas todas as conclusões, bem como respectivas contribuições deste trabalho num contexto actual, de forma a poder viabilizar projectos futuros.

Capítulo 2

Redes sem fios

2.1 Introdução

No ramo das telecomunicações, evolução é palavra de ordem. Temos assistido a um mundo globalizado onde as telecomunicações têm como missão aproximar pessoas, que, em inúmeros factos necessitam de aceder à informação através dum meio de comunicação à distância. Nesta área, as redes sem fios apresentam-se como uma solução inevitável. Graças a esta tecnologia é possível hoje nos nossos lares ter acesso à informação sem nos levantarmos do nosso sofá. Com um simples dispositivo que incorpore a norma 802.11 é possível acedermos a toda a informação disponível no maior repositório de informação – a Internet.

O IEEE é a instituição que rege e gere as normas necessárias para usufruirmos destes equipamentos. O IEEE é a maior organização profissional do mundo, sem fundos lucrativos. Foi fundado em 1963 através da fusão de duas grandes organizações existentes, o *American Institute of Electrical Engineers* (AIEE) e a *Institute of Radio Engineers* (IRE). O objectivo fulcral desta organização é promover o conhecimento nas áreas da electricidade, da electrónica e da informática. Objectivo este que conta o apoio de 375.000 membros de 160 países [**7**]. Esta organização, é sem dúvida, uma alavanca na evolução das comunicações, pois é responsável por muitas das normas dos dispositivos. Normas como o IEEE 754 (que possibilita maior precisão nos cálculos através da vírgula flutuante), o IEEE 1394, que corresponde ao *Firewire*) e, em destaque para esta dissertação,

a norma 802 (que se refere às redes locais e redes metropolitanas), são normas com um nome soante e importante na área da informática e das comunicações.

A norma 802.11 é um padrão internacional desenvolvido pelo IEEE que descreve as características de uma rede local sem fios. A designação Wi-Fi, corresponde a uma certificação industrial de produtos implementada a norma 802.11. Esta certificação tem como principal objectivo garantir a interoperabilidade entre diferentes dispositivos sem fios. A *Wi-Fi Alliance*, a antiga *Wireless Ethernet Compatibility Alliance* (WECA), foi o organismo responsável pela criação desta certificação [**8**].

2.2 802.11

Em 1989, a entidade americana responsável pela regulamentação do uso do espectro de frequências, a *Federal Communication Commission* (FCC), autorizou o uso de três faixas de frequência tendo em vista o uso futuro destas para comunicações sem fio. Um ano depois, o IEEE instaurou um comité, o *Wireless Local Area Networks Standard Working Group* (WLAN – SWG), para a definição de uma norma para a ligação sem fios. Em 1997, com sete anos de desenvolvimento e pesquisa nesta área, o comité responsável aprovou a norma IEEE 802.11 [**7**]. Começando com taxas de transferência na ordem máxima dos 2Mbps, esta norma sofreu uma evolução gigantesca, conseguindo em cerca de dez anos atingir uma taxa até 600 Mbps [**9**]. As principais preocupações no desenvolvimento desta norma foram o alcance/cobertura, o suporte de diversos canais, a sobreposição de várias redes, a robustez face às interferências, privacidade, segurança e o controlo de acesso ao meio.

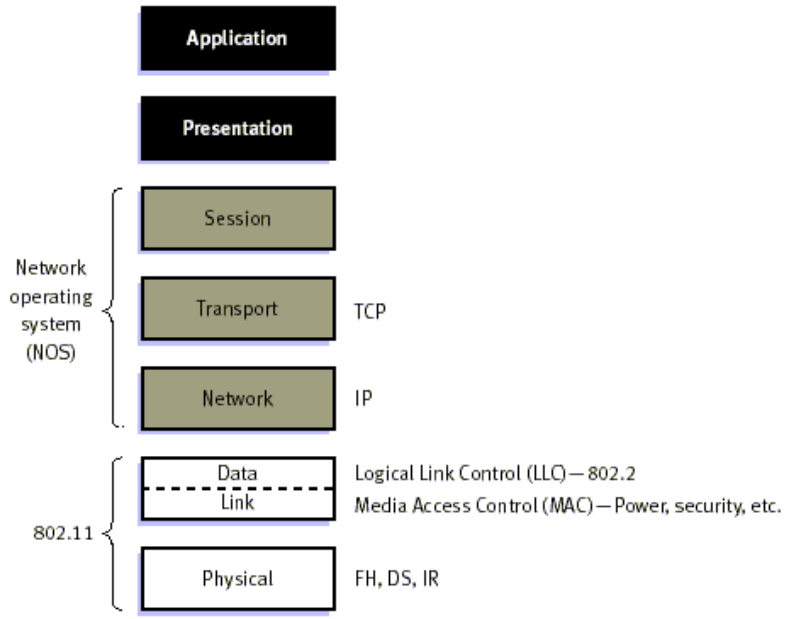

Figura 2.1 - Enquadramento do 802.11 no modelo OSI.

A norma 802.11 está indicada no modelo *Open Systems Interconnection* (OSI) da figura 2.1. Como é possível verificar, o 802.11 opera nas duas primeiras camadas do modelo OSI. Estas são responsáveis pela ligação física e pela ligação de dados às camadas superiores. Neste campo, o *Media Access Control* (MAC) é uma peça essencial, pois este protocolo é o gestor do acesso ao meio. Este controla quando uma estação pode ou não transmitir para outra estação [**10**]. O MAC 802.11 incorpora o *Distributed Coordinaton Function* (DCF), técnica que reduz o número e a frequência de colisões no meio. Esta técnica é baseada no mecanismo *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) [**11**]. O DCF é uma técnica que apresenta um algoritmo distribuído e que faz uso de dois *frames* de controlo: o *Request To Send* (RTS) e o *Clear To Send* (CTS) [**12**].

2.3 Evolução do 802.11

O 802.11 ao longo dos tempos, foi sofrendo constantes melhorias. Evolução na velocidade de transmissão, alcance e segurança, foram aspectos primordiais nesta evolução. Fazendo uma pequena selecção podemos indicar como especialmente interessantes:

2.3.1 802.11a

O 802.11a opera nos 5 GHz na banda do espectro. O uso desta frequência é pouco comum, o que faz com que o 802.11a seja mais vocacionado para sistemas privados de saúde, industriais e científicos, expondo-se a menos interferências. Esta evolução permitiu oferecer uma largura de banda até 54 Mbit/s e 4 canais não sobrepostos num total de 12 canais disponíveis na frequência que este opera no espectro [**13**].

2.3.2 802.11b

O 802.11b, apresentou-se o 802.11b como a primeira grande melhoria ao standard inicial 802.11: apresenta uma largura de banda até 11 Mbit/s e opera nos 2.4GHz. O uso de vários dispositivos, tais como bluetooth, telefones móveis e micro-ondas, criam uma interferência de grande peso nesta frequência em que opera o 802.11b. No entanto, o uso de uma frequência de fácil uso público e os baixos custos dos dispositivos que suportam esta norma apresentam-se como pontos muito fortes para a sua divulgação.

2.3.3 802.11g

O 802.11g é uma melhoria directa do 802.11b. Graças ao uso do sistema de modulação *Orthogonal Frequency Domain Multiplexing* (OFDM) foi possível nas mesmas frequências obter velocidades até 54 Mbit/s. Desta forma tornou-se concorrente directo ao 802.11a, onde apresenta as mesmas desvantagens que o 802.11b. Quanto a vantagens há a acrescentar o facto de suportar melhores normas de segurança, tais com o uso de autenticação e encriptação [**14**].

2.3.4 802.11n

Estudado durante seis anos e aprovado em 2009, o 802.11n apresenta melhorias significativas em termos de velocidade, com máximos até 600 Mbit/s [**9**]. O uso combinado do sistema *Multiple-Input Multiple-Output* (MIMO) com *Spatial Division Multiplexing* (SDM) permite o envio e recepção múltipla em vários canais, é o factor principal para o aumento significativo da velocidade máxima.

2.3.5 802.11s

O 802.11s é a norma responsável pelas redes em malha. Na arquitectura desta norma, o componente principal é o *Mesh Point* (MP) / *mesh station* (*mesh* STA). Ainda se encontra numa fase de normalização e de avaliação, mas pretende ser a norma que define como os *mesh points* se podem ligar em modo ad-hoc formando uma rede em malha sem fios [**15**] [**16**].

2.4 Espectro Electromagnético

Inicialmente, as redes sem fios foram licenciadas nos 2.4 GHz e 5 GHz, que eram associadas a bandas industriais, científicas e médicas – *Industrial, Scientific and Medical* (ISM) [**13**] [**17**]. Hoje, o mundo das redes sem fios, é dividida em quatro grandes variantes, 802.11a, 802.11b, 802.11g e o mais recente 802.11n. A co-existência entre eles existe e é permitida, mas existindo interferências entre eles, especialmente com o 802.11b e o 802.11g [**11**] [**12**] [**17**]. Os sistemas 802.11b e g, são responsáveis pelo uso do espectro dos 2.4GHz, usando tipos de transmissão diferentes das ondas rádio. O 802.11a e o 802.11n são os sistemas que actualmente se encontram associados aos 5 GHz de banda. Este último também se pode associar aos 2.4 GHz de banda.

A licença da banda dos 2.4GHZ estende-se por cerca de 83 MHz. Este excerto do espectro é, nas redes sem fios, dividido em canais separados 5 MHz como definido no standard do 802.11 [**14**] [**17**].

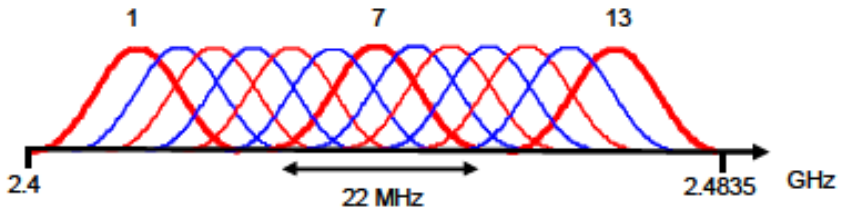

Figura 2.2 - Canais disponíveis para os dispositivos Wi-Fi que operam nos 2.4 GHz **[**17**]**.

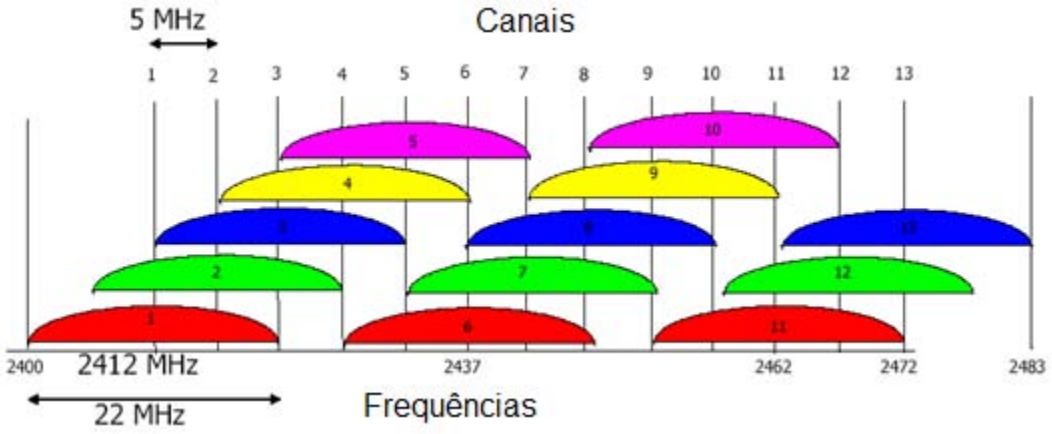

Figura 2.3 - Espaçamento entre os canais no espectro dos 2.4 GHz.

As figuras 2.2 e 2.3 mostram a divisão do espectro da banda dos 2.4GHz em pequenos canais. Como se pode verificar cada canal abrange cerca de 22 MHz, o que leva a sobreposição entre alguns canais, como é visível nas figuras 2.2 e 2.3. Existem poucos canais que não se sobreponham, tais como o 1, 7 e o 13 [**17**].

O número total de canais varia: na América são permitidos 11 canais, na Europa são permitidos 13 canais, e no Japão são permitidos 14 canais. Este último canal permitido centra-se nos 2483 MHz do espectro.

A sobreposição de canais leva a interferências rádio [**17**], o que influencia directamente o desempenho das redes sem fios, como será provado no capítulo 3. No entanto estas sobreposições têm impactos diferentes no 802.11b e no 802.11g. Esta diferença prende-se às diferentes formas como o sinal rádio é modulado. No sistema 802.11b é usado o DSSS, enquanto no sistema 802.11g é usado o sistema OFDM.

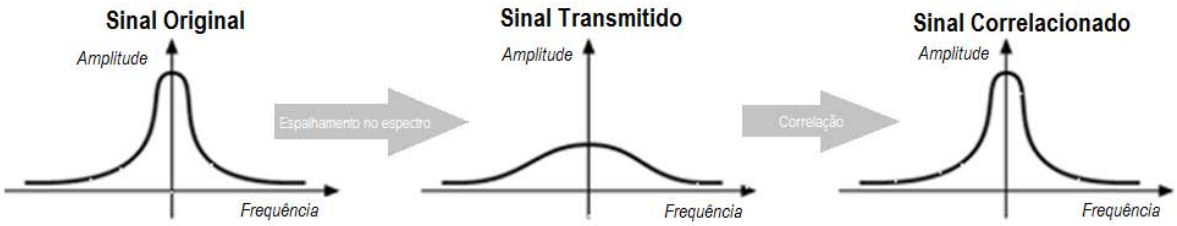

Figura 2.4 - Técnica de transmissão DSSS.

A figura 2.4, mostra a técnica de transmissão por sequência directa – *Direct Sequence Spread Spectrum* (DSSS). O DSSS é a técnica usada no 802.11b para transmissão do sinal, de forma controlada, partindo de uma banda larga de frequências. Este tipo de modulação foi preferido em favor do OFDM, aquando do aparecimento do 802.11g.

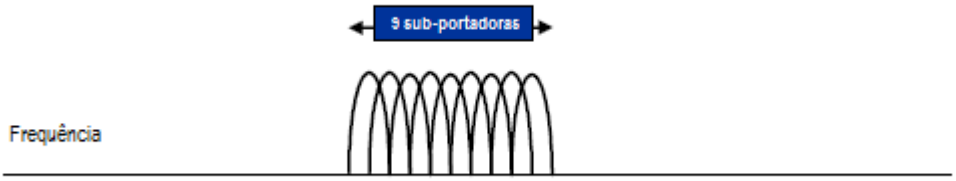

Figura 2.5 - Técnica de transmissão OFDM.

A figura 2.5, ilustra a técnica de transmissão OFDM com 9 sub-portadoras. Esta técnica multiplexada é baseada em *Frequency Division Multiplexing* (FDM). Aqui pretende-se o envio de múltiplos sinais em paralelo, onde cada sinal explora a sua gama de frequências, (denominadas sub-portadoras). É visível pela figura 2.5, que entre canais adjacentes existe sobreposição, podendo provocar interferências quando a separação entre canais é muito pequena.

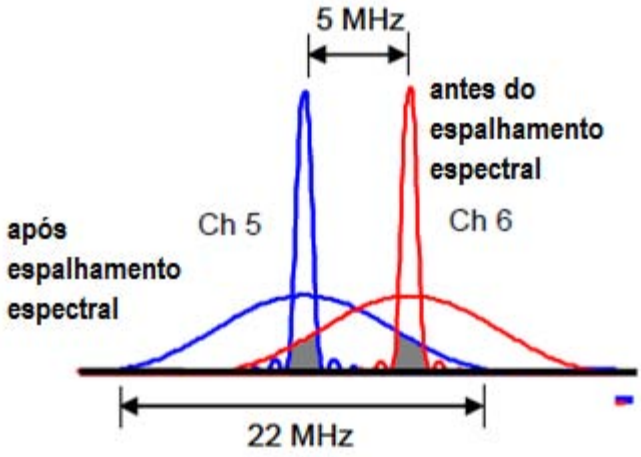

Figura 2.6 - Espectro de sinal de potência de dois sinais 802.11b em canais adjacentes **[**17**]**.

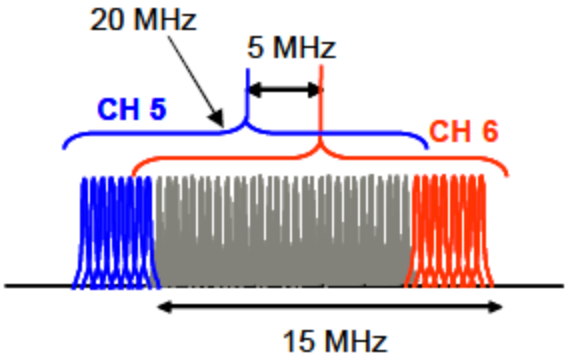

Figura 2.7 - Espectro de sinal de potência de dois sinais 802.11g em canais adjacentes **[**17**]**.

Nas figuras 2.6 e 2.7 são apresentadas diferentes sobreposições de canais adjacentes dos diferentes sistemas de transmissão, correspondentes aos sistemas 802.11b e 802.11g. Como se verifica pelas figuras 2.6 e 2.7, o OFDM do 802.11 g apresenta-se mais viável que o DSSS 802.11b, visto que o OFDM divide a informação a ser transmitida em pequeno conjuntos que são enviados simultaneamente, sendo também menos susceptível a interferências devido à forma como as frequências são utilizadas por esta técnica de transmissão. No entanto, atendendo a uma relação preço/desempenho o DSSS apresentou-se como a solução óptima para o uso doméstico e pequenas empresas. Por isso no sistema 802.11g na escolha de canal em que pretende operar é necessário ter em conta neles adjacentes, pois como se pode verificar pelas figuras 2.6 e 2.7, no sistema 802.11g, a sobreposição em canais adjacentes é maior no sistema 802.11g do que no sistema 802.11b [**18**]. Neste caso, a escolha deve remeter-se a canais não sobrepostos para que não existam interferências nas ondas rádio.

De forma geral, quando a separação é menor a 20 MHz, existe interferência em ambos os sistemas. Esta interferência provoca em qualquer um dos sistemas uma redução de desmpenho, concretamente, uma diminuição na largura de banda [**17**].

2.5 Arquitectura

No standard 802.11 a entidade básica é a estação – STA [**11**] [**14**]. A STA é um dispositivo que corre o protocolo 802.11, operando assim da forma descrita anteriormente nas duas primeiras camadas do modelo OSI. Um *Access Point* (AP) é um exemplo comum de uma estação. Para além desta entidade básica, existem quatro conceitos que norteiam a arquitectura do 802.11: *o*

Independent Basic Service Set (IBSS), o *Basic Service Set* (BSS), o *Extended Service Set* (ESS) e o *Distribution System* (DS).

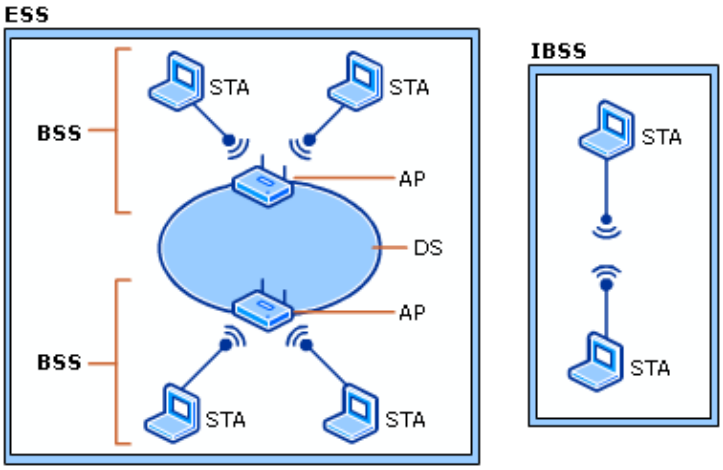

Figura 2.8 - Arquitectura do 802.11.

No 802.11 são definidos dois modos de funcionamento. Em ambos, existe um identificador da rede, um *Service Set Identity* (SSID). O SSID é periodicamente anunciado ao meio através dos *beacon frames* que são enviados pela estação responsável em cada modo. Os dois modos de funcionamentos são principalmente distintos pela presença ou ausência de um elemento central fixo. O modo ad-hoc é um sistema de redes sem fios distribuído, onde não há dependência de uma infra-estrutura de rede fixa; por outro lado, o modo infra-estruturado apresenta-se como um modo centralizado, onde existe uma infra-estrutura central que gera toda a rede fixa, denominado por estação base ou ponto de acesso.

A figura 2.8 mostra os elementos constituintes do 802.11. O BSS é a estrutura básica do 802.11. É composta por um ponto de acesso como suporte para um ou mais clientes sem fios. As estações, neste caso os clientes, comunicam sempre através do AP, e nunca directamente entre si. Os BSS podem ligar-se entre si através de um DS, formam assim um ESS [**19**] [**16**]. Usando a figura 2.8 como referência, o IBSS é uma rede sem fios onde são consideradas, no mínimo, duas estações e é usado na ausência de um DS. Este tipo de rede sem fios é vulgarmente conhecido por redes ad-hoc.

2.6 Redes Ad-hoc

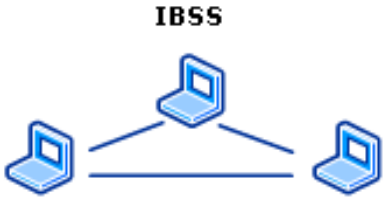

Figura 2.9 - Modo *ad-hoc* do 802.11.

O modo *ad-hoc* é representado na figura 2.9. É perceptível pela figura B que este modo opera sem nenhum ponto de acesso central, permitindo assim que os clientes/estações comuniquem entre si directamente. Na ausência de um AP, no modo *ad-hoc* (ou modo *peer-to-peer*) como não existe uma administração central, o primeiro cliente sem fios a formar o IBSS, assume algumas das responsabilidades de um AP, tal como o envio dos *beacons frames*, garantindo desta forma as ligações entre as estações através de mecanismos distribuídos de controlo e acesso ao meio [**14**] [**20**].

As redes ad-hoc viabilizam a comunicação entre estações fora da área de cobertura dos APs, sendo as estações intermediárias responsáveis pelo reenvio de pacotes. Assim estas apresentam uma grande facilidade de instalação, sem necessitarem de uma infra-estrutura prévia, com uma topologia dinâmica, bastando apenas determinar novas rotas e actualizar as rotas existentes. A mobilidade neste modo apresenta-se como uma vantagem primordial. Graças a protocolos de encaminhamento é possível expandir a área de cobertura garantindo conectividade entre todos os pontos. Este modo de funcionamento permite que falhas na rede possam ser facilmente recuperadas com uma reconfiguração da rede. No entanto, como existem limites na distância entre as estações, o processo de reconfiguração poderá ser dificultado. Este modo de funcionamento apresenta várias dificuldades: os protocolos de encaminhamento são um desafio para a comunidade científica, devido à mobilidade e à topologia dinâmica da rede descrita anteriormente; a taxa de erros neste tipo de rede é maior comparativamente à taxa de erros apresentada nas redes infra-estruturadas, provocando assim uma diminuição na largura de banda [**21**]. Apesar destas desvantagens, estas redes são bastante utilizadas em diversas aplicações, com necessidade de facilidade de instalação. Situações militares, situações de desastre e partilha de informações em conferências/reuniões são as principais aplicações para as redes ad-hoc.

2.6.1 Militares

O uso de redes ad-hoc para fins militares é cada vez mais usual. Graças às facilidades que este tipo de redes apresentam, é possível numa situação de ataque militar, sem nenhum tipo de infra-estrutura prévia, instalar uma rede ad-hoc e suportar toda uma operação militar.

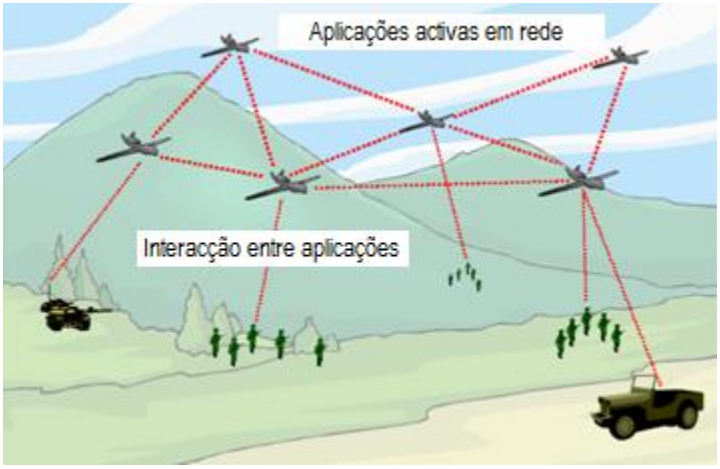

Figura 2.10 - Rede *ad-hoc* com fim militar **[**22**]**.

A figura 2.10 mostra um cenário de guerra onde está presente uma rede ad-hoc para uma operação militar. A rede é constituída por vários elementos, que se encontram em diferentes situações: aviões, carros de combate e soldados. Apesar de toda esta complexidade é possível criar este tipo de rede, com uma topologia dinâmica para que todos os elementos pertencentes à operação militar possam operar em comum.

Este tipo de utilização das redes ad-hoc está em crescente expansão, e faz parte da tecnologia de próxima geração que se pretende por em prática em situações militares.

2.6.2 Desastres

Numa altura em que vários tipos de ameaças são cada vez mais frequentes por diferentes motivos (tais como, ataques terroristas e assaltos, e desastres naturais que são uma constante preocupação, nomeadamente, furacões, terramotos e maremotos), o uso de redes ad-hoc em situações de desastre tornou-se uma necessidade. Desde o ataque terrorista nos Estados Unidos da América (EUA) a 11 de Setembro de 2001, houve uma maior preocupação em responder às necessidades apresentadas aquando desastres deste tipo [**23**]. Desta forma, planos de recuperação baseados em redes ad-hoc têm sido desenvolvidos e desta forma mais usados nos casos dos desastres.

O facto de se tratar de situações de emergência, implica uma falta de estrutura principal, à qual todos os elementos se conectam; por isso é usado o modo ad-hoc. Vários tipos de problemas da rede são próprios face à situação: custo, segurança, tempo de montagem, acessibilidade e a própria localização geográfica [**24**]. Mas um dos principais problemas com este tipo de redes é a topologia da rede que é apresentada em cada situação e em especial o desempenho da própria segundo essa topologia. Nunca é possível saber a priori a topologia. No entanto é possível idealizar algumas formas de resolver alguns problemas, desta forma obter soluções viáveis, isto porque em média é possível obter uma heterogeneidade na disposição dos nós [**25**].

Vários projectos estão a ser estudados e implementados nesta área, sendo de destacar o uso de motos equipadas de forma a funcionarem como nós de uma rede implementada em caso de desastre [**26**], e a vídeo vigilância e comunicação de um desastre através de nós agregados a balões [**27**]. A mobilidade e os custos reduzidos destes veículos motorizados são o factor principal para o uso dos mesmos neste tipo de redes. O uso de veículos motorizados equipados com um terminal próprio, uma antena e uma camera, é um protótipo que se encontra em estudo no *National Institute of Information and Communications Technology* (NICT) no Japão.

Figura 2.11 - Exemplo de um cenário de desastre [26].

Figura 2.12 - Protótipo desenvolvido [26].

As figuras 2.11 e 2.12, exemplificam um cenário de desastre e a utilização de veículos motorizados como nós da rede de forma a restabelecer as comunicações num dado local. Na situação da figura 2.11, as estruturas existentes apresentam-se danificadas, não podendo estabelecer comunicação. Então, com o tipo de equipamento representado na figura 2.12, é possível rapidamente montar uma rede ad-hoc que facilita o processo de resgate e recuperação da comunicação. Como os nós se podem encontrar em movimento, a topologia da rede é dinâmica o que poderá trazer alguns problemas com a taxa de erros na transmissão.

Também no Japão, mas na *Iwate Prefectural University* (IPU), uma equipa de engenheiros, construiu um balão que funciona como um nó de uma rede ad-hoc de vídeo vigilância.

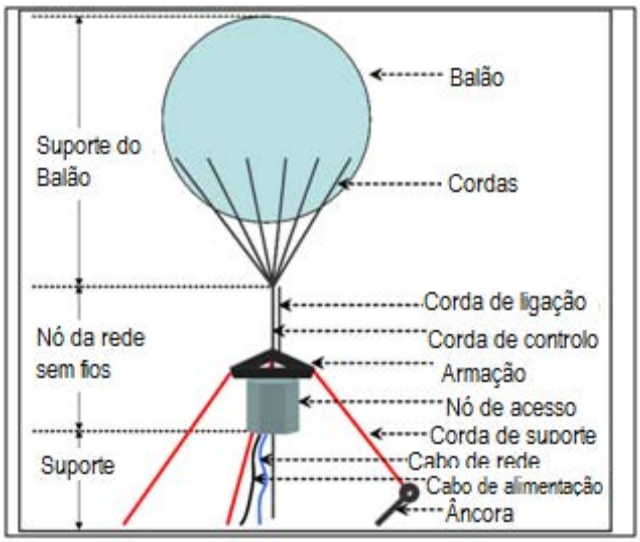

Figura 2.13 - Nó da rede agregado ao balão [27].

Figura 2.14 - Nó da rede agregado a um balão com um sistema de vídeo vigilância omnidireccional [28].

As figuras 2.13 e 2.14 representam um nó da rede ad-hoc para vídeo vigilância suportado por balão de ar. Este tipo de balões estando no ar compõem uma rede, que graças as cameras de vídeo incorporadas nos nós, permitem uma vídeo vigilância da área em necessidade.

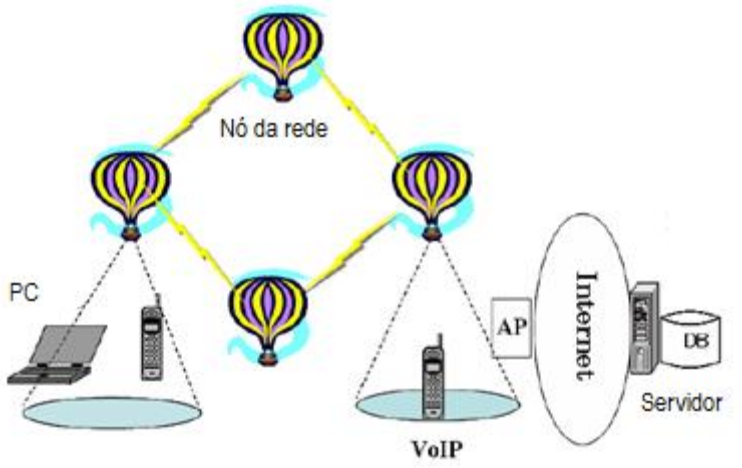

Figura 2.15 - Rede *ad-hoc* formada numa área desastre [28].

Na figura 2.15 está explicita a rede formada pelos nós (neste caso balões), que poderá ser implementada em situação de emergência. Esta abordagem encontra-se actualmente ainda em estudo não tendo ainda sido usados numa situação real. No entanto, apresenta-se como uma boa solução para um futuro próximo.

2.7 Redes infra-estruturadas

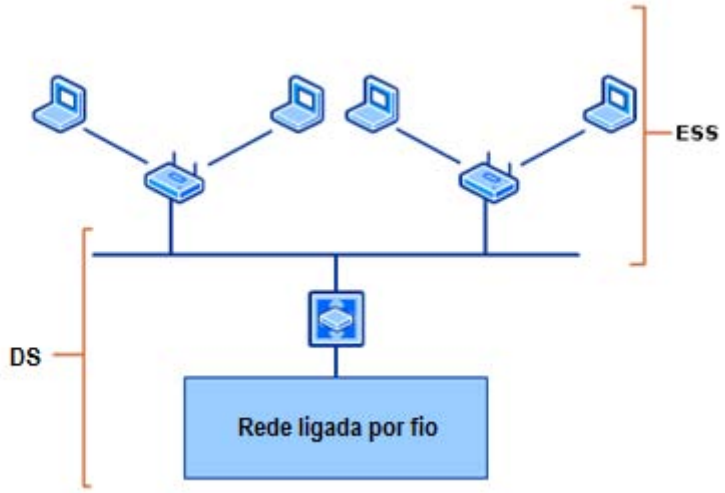

Figura 2.16 - Modo infra-estruturado do 802.11.

A figura 2.16 representa o modo infra-estruturado do 802.11. Este modo requer pelo menos um BSS. Como a figura 2.16 nos sugere, este modo apresenta uma administração central em cada BSS. Neste caso as estações não comunicam entre si, comunicam única e exclusivamente com os pontos de acesso. Os pontos de acesso pertencentes a cada BSS, servem como ligação para com a rede fixa, neste caso para com um DS. Existindo mais que um BSS, estes podem formar um ESS e desta forma interagir através do mesmo DS. O responsável pelo envio dos *beacon frames* é o AP de cada BSS [**14**].

O modo infra-estruturado, comparativamente ao modo ad-hoc, apresenta-se com algumas falhas. O facto de todas a estações estarem ligadas a um ponto em comum e não poderem comunicar entre si transporta consigo uma dependência que em caso de falha do ponto de acesso, provoca uma paragem total da rede.

As redes infra-estruturadas encontram-se em pequenos escritórios e casas domésticas assim como em grandes empresas. A necessidade de acesso à informação carrega consigo uma necessidade implícita da mobilidade. Desta forma existem hoje muitos *hotspots* onde se têm acesso à Internet (alguns grátis, outros pagos). No entanto existem redes que, patrocinadas por empresas ou por projectos das próprias localidades, oferecem através de uma rede infra-estruturada, uma ligação à Internet numa área considerável. As redes municipais são um exemplo deste conceito. Estas pretendem tornar uma cidade num enorme *hotspot.* A maior parte das cidades que oferecem este tipo de serviços encontram-se localizadas no continente americano. A cidade de Minneapolis é um exemplo de uma rede municipal paga.

Minneapolis é uma cidade do estado de Minnesota, dos EUA, onde em 2006, a *Universal Scientific Industrial* (USI*) Wireless Minnetonka* assinou um contrato válido por dez anos, de forma a tornar esta cidade uma das primeiras grandes cidades nos EUA com acesso a rede sem fios. Esta rede apresenta uma cobertura de aproximadamente 153 Km^2, contando com cerca de 2000 pontos de acesso sem fios, de forma a residentes, empresários e visitantes tenham acesso à informação em qualquer parte da cidade [**29**] [**30**].

Figura 2.17 - Rede sem fios na cidade Minneapolis [30].

A figura 2.17 apresenta a cobertura actual e futura da rede sem fios implementada na cidade de Minneapolis. Esta rede, segundo a operadora responsável, pretende até final de 2009 cobrir toda a área projectada. Sendo um projecto de carácter público, apresenta-se como uma mais-valia no acesso à informação para a cidade. Apesar de ser pago, existem limitações no serviço prestado: o facto dos pontos de acesso se encontrarem essencialmente em locais exteriores às casas de habitação dificulta o acesso a moradores, pois o sinal apresenta-se fraco no interior das habitações. No entanto, a cidade é receptiva ao serviço e aceita com bom grado esta rede estendida pela cidade.

Existem nos EUA, várias outras cidades com redes municipais implementadas, e muitas a trabalhar nesse sentido. A gigante *Google* quer tornar-se num potencial investidor neste tipo de redes, inaugurando a *Google Wi-Fi.* Actualmente oferece um serviço gratuito de acesso à internet apenas em *Mountain View*, no estado da Califórnia, mas pretende estende-lo nos próximos tempos a várias cidades do mesmo estado, Califórnia, em particular para a cidade de *San Francisco* [**31**]. A *Google* oferece um serviço a cerca de 72.000 residentes fixos, não contando com os visitantes da cidade, que usufruem também das instalações, mas não como residentes. É de destacar a cobertura de cerca de 31 Km^2, mas apresentando locais onde ainda não foi possível garantir conectividade ao serviço. No entanto, servindo como primeiro modelo para as redes infra-estruturadas da *Google,* apresenta-se como um bom serviço público com boas perspectivas de expansão para outras cidades.

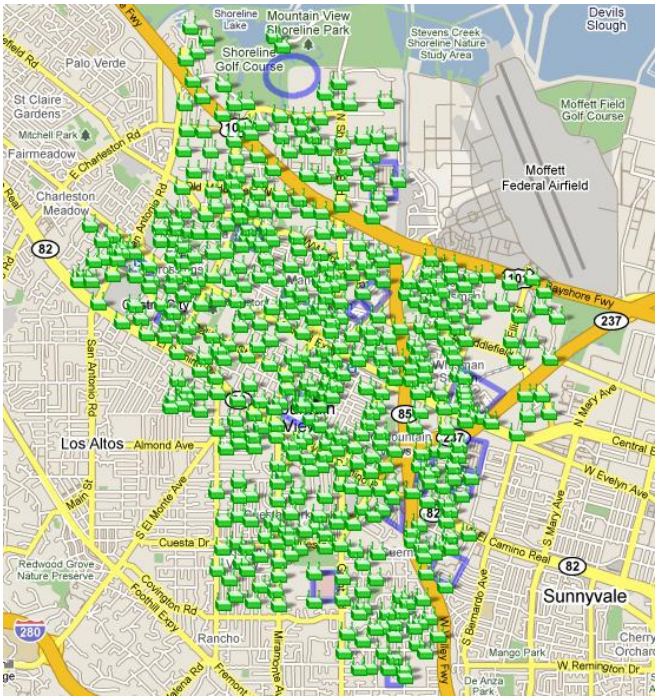

Figura 2.18 - Rede sem fios na cidade Mountain View **[**31**]**.

A figura 2.18 representa a cobertura da *Google Wi-Fi Mountain View*. Para suporte a esta cidade foi necessário um número elevado de pontos de acesso, como se pode verificar pela figura. Uma cobertura alargada, como apresentada na figura 2.18, representa para o utilizador a mobilidade no acesso à informação, factor importante nos dias de hoje.

Nesta secção foram apresentadas as redes infra-estruturadas. Seguidamente, serão introduzidas as redes em malha sem fios, redes estas que operam de forma diferente, mas que poderão ter objectivos em comum com as redes já aqui apresentadas.

2.8 Redes Mesh Sem Fios

Os dois modos apresentados anteriormente, ad-hoc e infra-estruturado, são modos nativos do 802.11. No entanto, mais recentemente, foi desenvolvido um terceiro modo infra-estruturado, mas onde as estações operam em modo ad-hoc. Este modo apresenta-se como uma junção dos dois modos nativos: as estações funcionam como emissores e receptores não só para os clientes, mas também entre elas próprias. Estas redes designam-se por *wireless mesh networks* (WMN). Podem ter um papel importante no mundo actual: são usadas já nos campos de batalha, em operações militares; estão a ser implementadas de forma que os contadores comuns de água das casas de habitação consigam comunicar entre si, permitindo efectuar uma contagem remotamente, sem deslocação de um técnico para a contagem; no programa mundial "*one laptop per child*" pretende-se que as crianças, através de uma rede sem fios em malha formada através dos portáteis, possam trocar informação e aceder à internet [**32**]; são ainda usadas para monitorização remota da meteorologia [**33**].

Nesta área existem combinações com outras tecnologias que poderão apresentar-se num futuro próximo como soluções puras, mais concretamente soluções de redes sem fios WiFi com a tecnologia *Wireless Interoperability for Microwave Access* (WiMAX) [**34**]. Serão seguidamente apresentadas as tecnologias disponíveis com exemplos reais.

2.8.1 802.11

Usando o 802.11 é possível construir uma rede sem fios em malha. A maior desvantagem na adaptação destes recursos para as WMN, é o facto do comportamento

destas com os protocolos de encaminhamento existentes não ser o mais desejado e as interferências influenciarem a largura de banda disponível para os clientes. Daí houve necessidade de criar um novo padrão específico para este tipo de redes, o 802.11s [**15**].

Figura 2.19 - Exemplo de uma rede sem fios em malha (WMN) **[**35**]**.

O modo de funcionamento deste tipo de redes está representado na figura 2.19. É visível que neste tipo de redes se pretende cobrir a maior parte de zona metropolitana de uma cidade ou de uma dada região. Segundo os especialistas *Netkrom* [**35**], as estações são implementadas no cimo dos edifícios conseguindo assim fornecer os seus serviços aos prédios e aos clientes que se encontram mais próximos do solo. Um exemplo deste tipo de serviços encontra-se na Austrália e nomeia-se por ***metro****mesh* [**36**]. Apresenta diferentes tarifas dependendo dos vários serviços oferecidos.

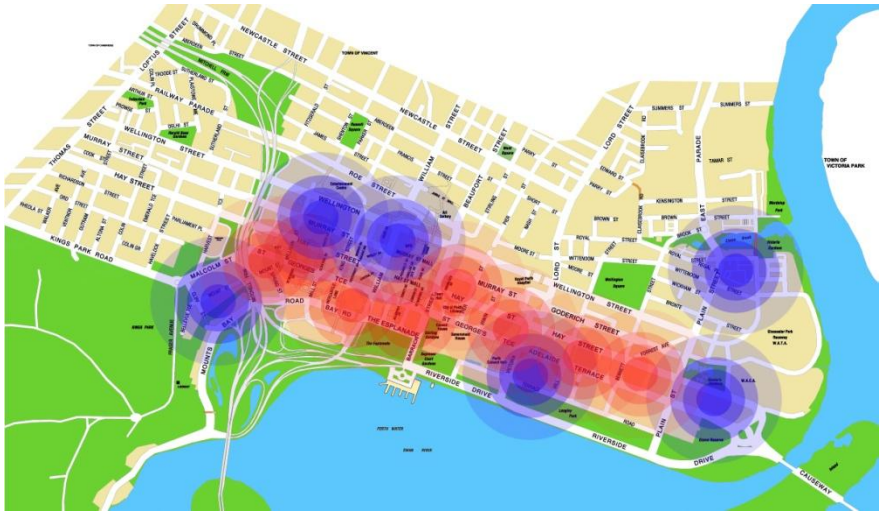

Figura 2.20 - Rede sem fios em malha ***metro****mesh* [36].

A figura 2.20 ilustra à cobertura da ***metro****mesh,* na cidade de Perth, Austrália. Esta rede pretende servir toda a população, residente ou não, com os seus serviços, apresentando assim a cobertura representada no mapa da figura 2.20. Este é um exemplo real deste tipo de redes, no entanto existe a possibilidade de combinar várias tecnologias de forma a obter melhores alcances com melhores larguras de bandas, isto é, melhores condições para os clientes. O caso mais conhecido é a junção do 802.11 com o 802.16.

2.8.2 802.11+802.16

O WiMAX, IEEE 802.16, é uma tecnologia que opera nas frequências dos 2GHz e 11 GHz, e que ainda se apresenta num processo de amadurecimento. Tem uma arquitectura centralizada fornecida pela *Base Station* (BS) central que tem elementos associados, denominados por *Subscriber Stations* (SS). Com esta BS pretende-se obter um melhor desempenho de comunicação usando recursos mais recentes. A coexistência

da interoperabilidade das duas normas existe (802.11 e 802.16); no entanto, esta coexistência não é um problema de resolução fácil [**34**].

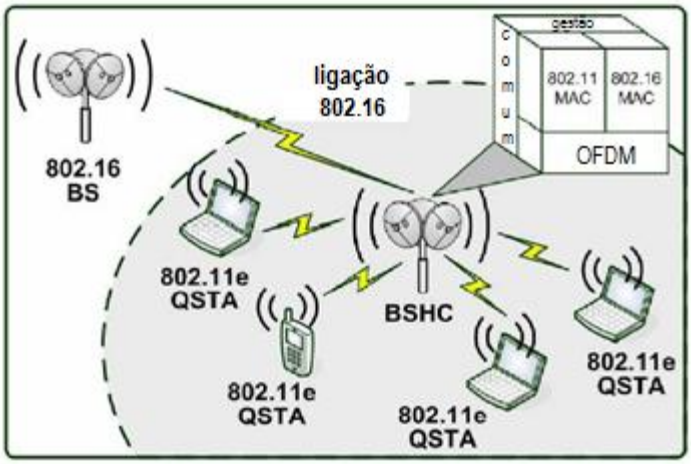

Figura 2.21 - Cenário da coexistência do 802.11 com o 802.16 **[**34**]**.

A figura 2.21 representa um cenário simples da coexistência com as duas normas: 802.11 e 802.16. As normas apresentam diferenças significativas na camada MAC: o 802.16 apresenta-se centralmente coordenada pelo seu protocolo MAC, o 802.11 apresenta um controlo distribuído e um acesso ao meio controlado. No entanto, o facto que usarem o OFDM como a tecnologia de transmissão permite a sua coexistência. Esta integração implica a existência de um protocolo onde os dois possam operar [**34**]. Nesse caso, como se verifica na figura 2.21, existe uma estação central com um *hybrid coordinator* que permite a interoperabilidade dos dois padrões.

O uso combinado de tecnologia por vezes não se torna tão simples de implementar; o cenário apresentado na figura 2.21, é um exemplo de uma rede simples onde há necessidade de um intermediário que consiga transpor a informação de uma tecnologia para a outra. Por isso, não existe nenhuma rede actualmente que se apresente completamente implementada com o uso do 802.16 como tecnologia auxiliar. No entanto existem testes com bons resultados mas com algumas adversidades expostas na coordenação dos standards [**34**]. O facto da tecnologia 802.16 só recentemente ter atingido a versão final é o maior problema nesta coexistência; no entanto esta combinação tem mesmo agora apresentado resultados interessantes [**37**].

2.8.3 802.11s

Com a necessidade de criar uma norma que suportasse e especificasse todos os requisitos das redes sem fios em malha, um grupo do IEEE tratou do desenvolvimento de um novo standard, o 802.11s [**38**]. Os estudos acerca deste tipo de redes têm aumentado e formou-se um consórcio de empresas que estão dispostas a financiar e a colaborar directamente com a criação de uma implementação *open source* do IEEE 802.11s *wireless mesh standard* [**39**].

O facto de o espectro dos canais das redes sem fios serem limitados e a potência de transmissão não pode exceder os 100mW, limita o alcance disponível por uma rede sem fios, apesar de existirem com a norma 802.11n, taxas de transferência até 600 Mb/s. Assim todos os constituintes desta norma ligam-se sem fios, formando assim uma rede sem fios em malha [**16**].

Esta norma rege-se pela arquitectura do 802.11, mas com algumas adaptações. Enquanto no 802.11 a entidade elementar é uma estação, no 802.11 é o *mesh point*; desta forma, estes têm a capacidade de trocar informação entre si. No mesmo sentido comparativo, no 802.11 existe o BSS, que no 802.11s é analogamente um *mesh* BSS. Tendo em conta que se trata de uma rede sem fios em malha, à partida todos os MP têm acesso entre si, bastando apenas uma *mesh path* de todos os MPs constituintes da rede. A maior adversidade que se apresenta ainda neste momento é o acesso ao meio que apresenta reduções de desempenho consideráveis, por dificuldades com o meio rádio [**15**]. Assim como no 802.11 o MAC resolve os problemas de acesso ao meio, no 802.11s o *Mesh Deterministic Access* (MDA) procura dar resposta a estes problemas.

Tal como no modo *ad-hoc*, o uso das redes sem fios em malha é mais procurado com fins militares. Neste sentido a MITRE Corporation, uma organização sem fins lucrativos que fornece serviços para o governo [**40**], tem participado nos encontros do IEEE 802.11 de forma a contribuir em particular para o standard 802.11s. A participação activa da MITRE nos encontros influenciou de forma activa que os requisitos governamentais passassem a ser incluídos nos standards [**41**].

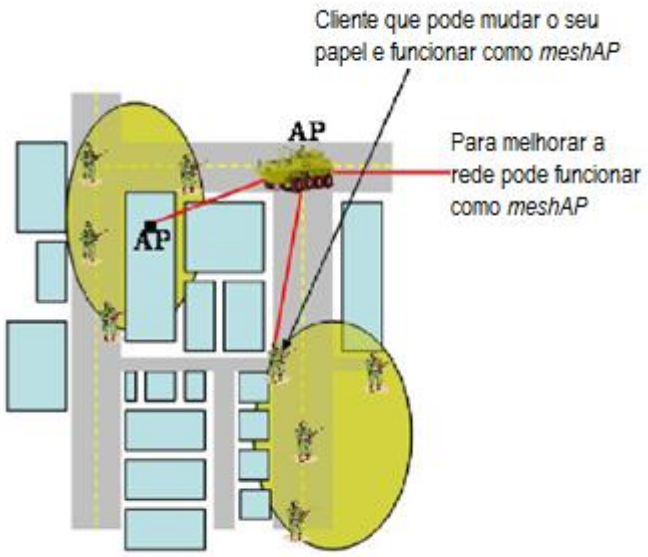

Figura 2.22 - Cenário do uso militar do 802.11s **[**41**]**.

A figura 2.22, mostra o exemplo de um cenário militar. Com a imagem representada na figura 2.22, a MITRE classifica dois tipos de utilização no uso militar de redes sem fios em malha: a primeira categoria, não modo de combate; e a segunda categoria, modo de combate. As duas categorias distinguem-se pela mobilidade dos nós, por uma auto gestão da rede, onde o consumo eléctrico é um aspecto relevante.

Em geral, o 802.11s apresenta-se como um standard de futuro, pois necessita ainda de evoluir em vários aspectos. No entanto, cada vez mais estudos e testes estão a ser efectuados, para que num futuro próximo este standard possa ter um papel mais activo na sociedade.

2.9 Sumário / Conclusões

No capítulo dois foram resumidos aspectos relevantes do standard IEEE 802.11 para esta dissertação. Os aspectos teóricos neste capítulo foram expostos de forma aos próximos capítulos 3 e 4 se apresentarem como uma confirmação prática de vários aspectos práticos referenciados. Por fim, foi possível obter uma apreciação global dos modos de funcionamento do 802.11 e perspectivas de futuro, tendo em conta também outras tecnologias.

Capítulo 3

Estudo de densidade de terminais

3.1 Densidade de terminais

A massificação do uso das redes sem fios apresenta desvantagens para o desempenho das redes. A necessidade que os utilizadores têm no acesso à informação é a maior causa para uma crescente existência de dispositivos que incorporam a norma 802.11, em especial as normas 802.11a, 802.11b e 802.11g. Este aumento no número de pontos de acesso em certos locais pode levar para uma saturação do meio onde se encontram, provocando assim uma ineficiência dos mesmos. Por vezes pode ser impossível a conexão [**10**] [**11**].

Nesta área podem ser realizados inúmeros testes de desempenho, relacionados de várias formas com factores reais e com a qual é preciso contar no estudo. Neste estudo, pretende-se saber quais as influências na transmissão efectiva de dados que as interferências nas ondas rádios provocam, num cenário com múltiplos pontos de acesso.

Este estudo apresenta dois pontos primordiais para esta dissertação: primeiro uma percepção prática dos problemas das interferências nas ondas rádio apresentados no capítulo anterior, e em segundo como complemento para aquisição de conhecimentos necessários para o próximo capítulo, núcleo desta dissertação.

3.2 Descrição

Este estudo será seguidamente descrito neste capítulo. O estudo de suporte às normas IEEE 802.11b/IEEE 802.g pretendia analisar a interferência de uma densidade de terminais no desempenho de duas redes pretendidas para estudo.

No estudo, existem quatro intervenientes: pontos emissores de informação (pontos de acesso 802.11g), clientes receptores dessa informação, um meio saturador para a transmissão e um analisador do meio. No total, recorreu-se a 22 APs e três computadores clientes, funcionando um deles como analisador.

Os pontos de acesso usados foram equipamentos FON 1.0 com o DD-WRT. Foram mantidas todas as configurações gerais de fábrica que são instaladas pelo DD-WRT, apenas alterando o **SSID** e o **canal** onde operavam. Neste caso:

- Havia dois pontos de acesso emissores: o primeiro com SSID **dd-wrt** e o segundo com SSID **dd-wrt_ap2**, encontravam-se também configurados para operarem no canal 6 (2437Mhz).
- Os restantes vinte pontos de acesso foram configurados conforme os cenários que serão posteriormente descritos.
- Não foi usada encriptação ou qualquer método de segurança; e o tempo de envio de *beacon* frames é o padrão.

Os canais escolhidos nos terminais para saturação do meio, foram o canal 1, na frequência dos 2.437 GHz e o canal 6, nos 2.412 GHz. Tendo em conta a sobreposição de canais, escolheram-se estes dois canais, pois esta sobreposição assim não se verifica [**17**].

3.2.1 *Hardware/Software* Utilizado

Pontos de acesso	
Marca / Versão	**FON 1.0**
Memória	Flash 8 MB / SDRAM: 16 MB
Ficha da antena	RP-SMA
Antena	Omnidireccional (2dBi)
Alimentação:	Entrada: 100-240V ~ 50-60Hz 0.3A Saída: 5V, 2.0A DC
Consumo	4 Watt
Compatível com as normas IEEE 802.11b/802.11g	

Software	
DD-WRT	É um *firmware* grátis e de código aberto, baseado em Linux, que opera com a maior parte dos pontos de acesso 802.11a/b/g. A facilidade de utilização e configuração, é um dos pontos fortes deste *software*.

Tabela 3.1 - Especificações técnicas dos pontos de acesso

Computadores Clientes		
	PC1	**PC2**
Marca	Compal	Acer
Modelo	FL-90	Ferrari 3000
Processador	Intel Core 2 Duo T7700	AMD Athlon XP 2500+
Memória	4096 RAM DDR2	1280 RAM DDR

Placas *Wireless*	
Marca	Netgear
Modelo	WG511T
Velocidades Suportadas	1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54 e 108Mbps
Tipos de modulação	OFDM com BPSK, QPSK, 16QAM, 64QAM, DBPSK, DQPSK e CCK
Frequências	2.412 ~ 2.472 GHz

Tabela 3.2 - Especificações técnicas dos computadores clientes

Em ambos os computadores, o sistema operativo utilizado foi o Ubuntu 8.04 *Long Term Support* (LTS) Desktop Edition [**42**] com o uso de uma ferramenta de análise do desempenho da largura de banda, o *Iperf* [**43**].

Analisador	
Marca	Compal
Modelo	HEL-80
Processador	Intel Core 2 Duo T7200
Memória	2048Mb RAM DDR2
Placa *Wireless*	Intel PRO/*Wireless* 3945 ABG
S.O.	Windows XP Professional SP3
Software	
NetStumbler v. 0.4.0	*Scanner* de redes sem fios para as normas 802.11b e g [**44**]
Wi-Spy 2.4x com Chanalyzer v. 3.1 [45**]**	Pequeno dispositivo USB desenhado para capturar toda actividade rádio no espectro dos 2.4 GHz. Este dispositivo é acompanhado pelo Chanalyzer, programa que mostra a actividade capturada.

Tabela 3.3 - Especificações técnicas do analisador

Para analisar o meio foi usado Chanalyzer e o NetStumbler, obtendo assim o resultado do comportamento do meio, com a densidade dos terminais. Ao servidor, estavam ligadas as duas estações emissoras, à qual para cada uma encontrava-se um servidor do *Iperf*.

Servidor	
Processador	AMD Athlon XP 2100+
Memória	512Mb RAM DDR
Software	
S.O.	Ubuntu 8.04 LTS Desktop Edition [**42**]
Iperf	Dois servidores *Iperf* para cada uma das estações emissoras a correr em processos diferentes com portas diferentes

Tabela 3.4 - Especificações técnicas do servidor

Este estudo foi realizado numa casa privada, onde foram garantidas condições necessárias para a concretização do mesmo.

Figura 3.1 - Densidade de terminais no local de estudo.

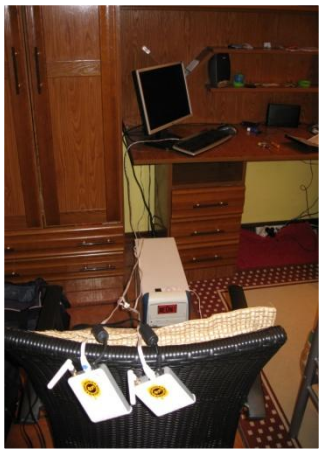

Figura 3.2 - Pontos de acesso 1 e 2 no local do estudo.

As figuras 3.1 e 3.2, mostram o local onde o estudo foi realizado. Na figura 3.1, são visíveis os pontos de acesso dispostos ao longo da mesa de trabalho e o analisador. Na figura 3.2 são apresentados os pontos de acesso emissores ligados ao servidor.

3.3 Cenários

Foram considerados três cenários de estudo, e em cada um dos cenários foram considerados quatro configurações diferentes. Os cenários dizem respeito à rede formada pelos computadores clientes. As configurações estão relacionadas aos APs saturadores do meio.

Para cada um dos cenários foram corridas quatro testes, correspondentes às quatro configurações distintas existentes. Os cenários e as configurações de cada uma das simulações encontram-se descritos seguidamente.

Cenário A

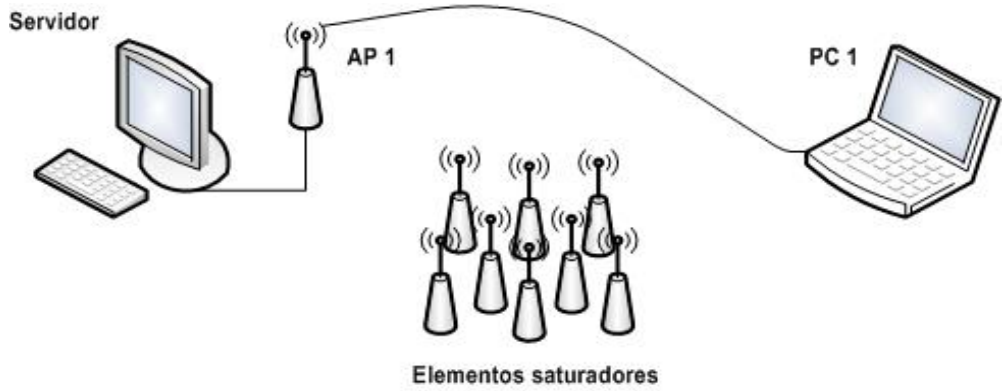

Figura 3.3 – Representação do cenário A.

A figura 3.3 representa o cenário A. Este cenário é composto pelo servidor, pelo AP1, pelo PC1 e os APs saturadores do meio.

Cenário B

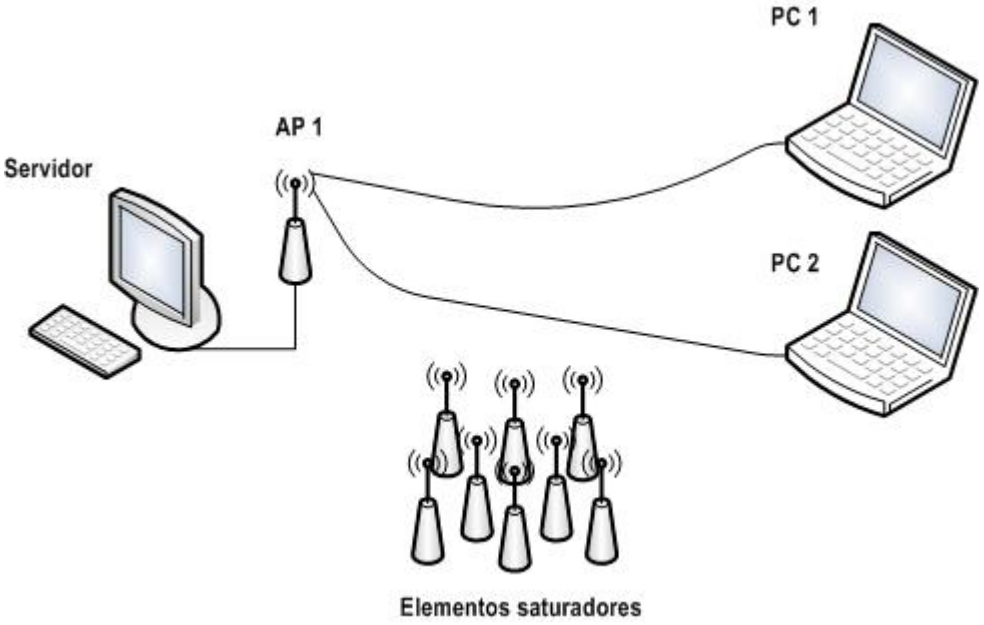

Figura 3.4 - Representação do cenário B.

A figura 3.4 corresponde ao cenário B. Neste cenário, o servidor, o AP1, o PC1 e o PC2 são os intervenientes.

Cenário C

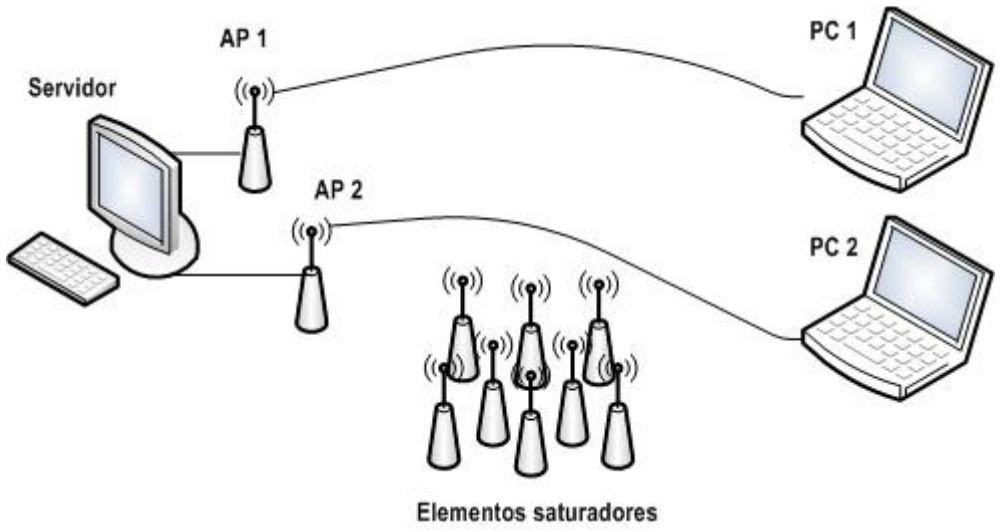

Figura 3.5 - Representação do cenário C.

A figura 4.5 refere-se ao cenário C. Neste cenário o servidor, o AP1 e 2 e os PC 1 e 2 são os elementos constituintes.

A transmissão de informação era, para todos os cenários, do servidores para os seus clientes, PC1 e/ou PC2, e vice-versa. Estes dados eram recebidos ou enviados pelo(s) AP(s) correspondentes (AP1 e/ou AP2).

3.4 Configurações

Neste estudo, a simulação foi repartida em cenários e em configurações, para que abrangesse o maior número de factores interventivos no meio. Desta forma foram consideradas quatro configurações distintas para cada cenário.

	SSID	CANAL	Simulações	Observações
I	dd-wrt	6 – 2.437GHz	1, 2, e 3	-
II	dd-wrt***X***	6 – 2.437GHz	4, 5 e 6	***X*** corresponde ao número do AP, [1-20]
III	dd-wt***X***	1 – 2.412GHz; 6 – 2.437GHz	7, 8 e 9	***X*** corresponde ao número do AP, [1-20]
IV	dd-wrt	1 – 2.412GHz; 6 – 2.437GHz	10, 11 e 12	-

Tabela 3.5 - Tabela explicativa das configurações.

A tabela 3.5, agrupa dois factores, o SSID e o canal, em várias configurações. Demonstra também a correspondência das mesmas com as simulações. Desta forma é possível verificar que o estudo se encontra agrupado em conjuntos de três simulações, visto que para cada configuração, existem três cenários distintos.

3.5 Objectivos

Saturando um meio onde se pretende que haja transferência de informação de um AP para um cliente, pode-se conseguir saber qual o comportamento desta transferência na presença de uma grande densidade de terminais.

A interferência provocada pelas ondas rádio na comunicação é o factor em destaque neste estudo. Com analisadores do sinal rádio emitido, pretende-se medir a quantidade de ruído que é introduzido quando ocorrem transferências num meio saturado.

Com os resultados obtidos, pretende-se adquirir conhecimentos que possam contribuir para a componente mais prática desta dissertação, nomeadamente o capítulo 4.

3.6 Metodologia

Os pontos de acesso FON 1.0, apresentam um *firmware* próprio, *firmware* este que não responde na sua totalidade às necessidades deste estudo. Visto o DD-WRT ser um *firmware* grátis, estável, de fácil instalação e configuração, optou-se por este último. A instalação deste *firmware* foi efectuada através da ligação directa por cabo entre uma porta RS232 e os pinos específicos na FON 1.0, e procedendo às configurações necessárias através da consola, mais propriamente o Putty [**46**].

Após instalação do DD-WRT em todos os pontos de acesso, foi necessário proceder à configuração dos mesmos conforme foi explicado anteriormente. Para além dos pontos de acesso, os clientes necessitaram de alguma configuração extra. A associação do cliente ao ponto de acesso desejado era necessário ser garantido para que num meio saturado com vários pontos de acesso com o mesmo SSID, o cliente estabelece-se a ligação com o ponto de acesso correcto. Por isso, foi necessário que os clientes se associassem através do endereço físico do ponto de acesso. Com este tipo de associação é possível evitar erros no *Iperf* do tipo *"broken pipe"*.

Para execução da simulação foram agregados todos os APs em conjuntos de cinco e numerados. Esta regra apenas serviu para que as alterações das configurações no DD-WRT fossem mais fáceis.

O funcionamento da ferramenta *Iperf* é um aspecto a referir. Esta funciona segundo uma arquitectura servidor-cliente, e para tal é necessário configurar ambas as partes. Para que os pontos de acesso funcionassem como servidores, foi necessário ligar os mesmos a um computador, através de um cabo de rede, para que este pudesse correr o *Iperf* em modo servidor. Em portas diferentes é possível correr este modo e estabelecer uma ligação por cada servidor configurado na porta correcta.

O uso de placas de rede sem fios iguais nos clientes é um factor importante, pois assim é possível ter uma simulação mais próxima da realidade e homogénea, apesar de os componentes dos computadores clientes serem diferentes. Outro ponto a notar é o facto de as distâncias dos clientes aos pontos de acesso poderem variar alguns centímetros, podendo afectar os resultados finais; no entanto tentou minimizar-se esse efeito.

O meio saturador foi alterado através de uma densidade de terminais, neste caso até no máximo de 20 pontos de acesso. A densidade dos terminais no meio foi elevada gradualmente; neste caso, o número de terminais ligados aumentava cinco em cinco durante a simulação.

Em cada teste, foram feitas três medições, de forma a usar numa comparação *a posteriori* os valores médios.

3.7 Resultados

Primeiro foram registados valores iniciais para posterior análise, comparação e estudo dos resultados. Estes valores não se encontram associados a um meio saturador associado, isto é, apenas os emissores e receptores de informação se encontram ligados para este teste de desempenho.

Cenário	PC1	PC2
A	24,3 Mbits/sec	-
B	12,0 Mbits/sec	13,1 Mbits/sec
C	11,4 Mbits/sec	10,2 Mbits/sec

Tabela 3.6 - Valores referência obtidos inicialmente.

A tabela 3.6 refere-se aos valores iniciais do estudo. Estes valores apresentam-se como valores de referência para o mesmo, dado que, neste caso não está envolvido nenhum elemento saturador. Com estes dados será possível comparar quanto um elemento saturador reduz o desempenho de uma rede.

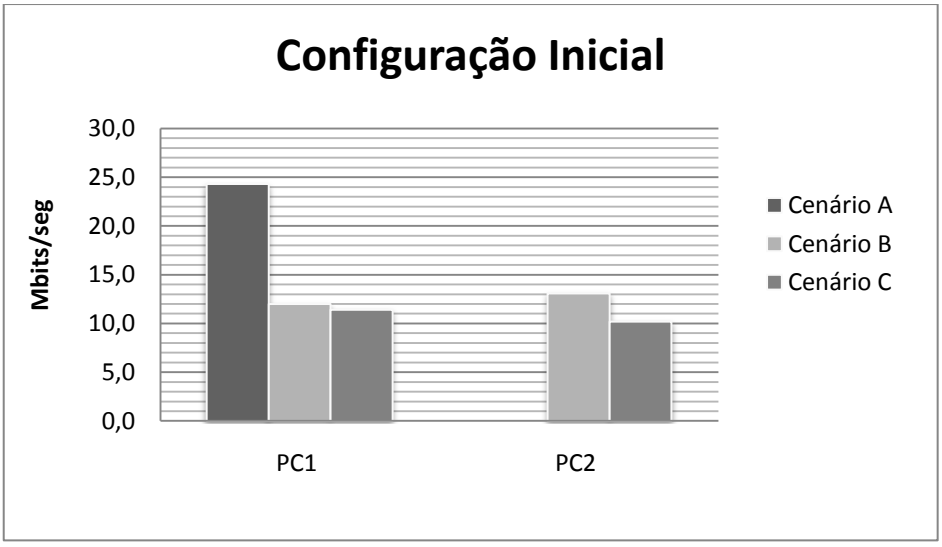

Figura 3.6 - Valores iniciais para todos os cenários no PC1 e PC2.

A figura 3.6 mostra-nos os valores de referência considerados para este estudo. É possível verificar que existe um decréscimo em todos os cenários. Este decréscimo era esperado pelo facto de inicialmente apenas existir um único cliente ligado a um AP e, no segundo cenário existirem dois clientes conectados a um AP, o que reduz o desempenho. No terceiro cenário, apesar de existirem dois clientes e dois APs ligados separadamente, este contribuem para um maior número de pacotes no meio envolvente, provocando colisões e desta forma reduzirem o seu desempenho.

Nesta fase todos os resultados serão apresentados e comentados resumidamente por ordem crescente das configurações, isto é, para cada configuração será apresentado o resultado comentado de cada um dos cenários. O *throughput* em relação ao SSID, canal e número de pontos de acessos ligados no meio será o ponto de comparação entre todas as configurações, e também entre os cenários. Tal como na figura anterior, nas figuras 3.7,3.11,3.15 e 3.19 existem sempre dois conjuntos de valores referentes ao PC1 e ao PC2 pela ordem indicada, em relação ao eixo horizontal (neste caso, ao eixo que indica a densidade dos pontos de acesso).

Para cada configuração, para além do gráfico geral, é depois apresentada e descrita para cada cenário, a linha de tendência dos valores obtidos experimentalmente.

3.7.1 Throughtput vs Configuração I

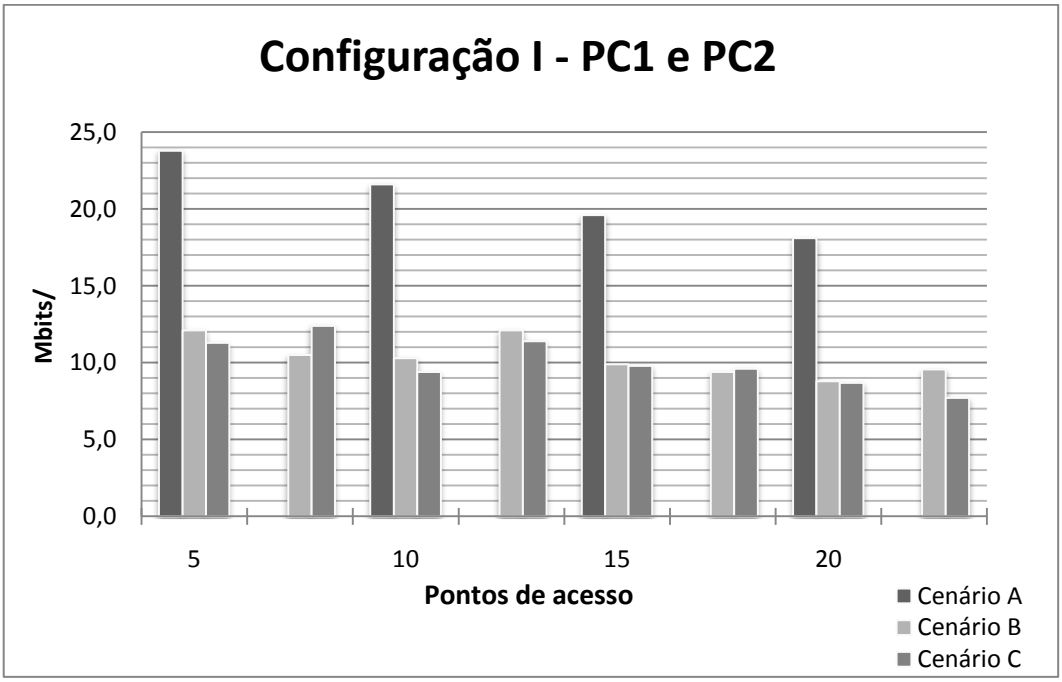

Figura 3.7 - Valores referentes à configuração I.

A figura 3.7 é um gráfico com a representação de todos os cenários para a configuração I, relacionados com o aumento de pontos de acesso como elemento saturador. De uma forma geral, verifica-se uma diminuição da largura de banda em todos os casos no PC1 e no PC2 correspondente à introdução gradual de pontos de acesso saturadores.

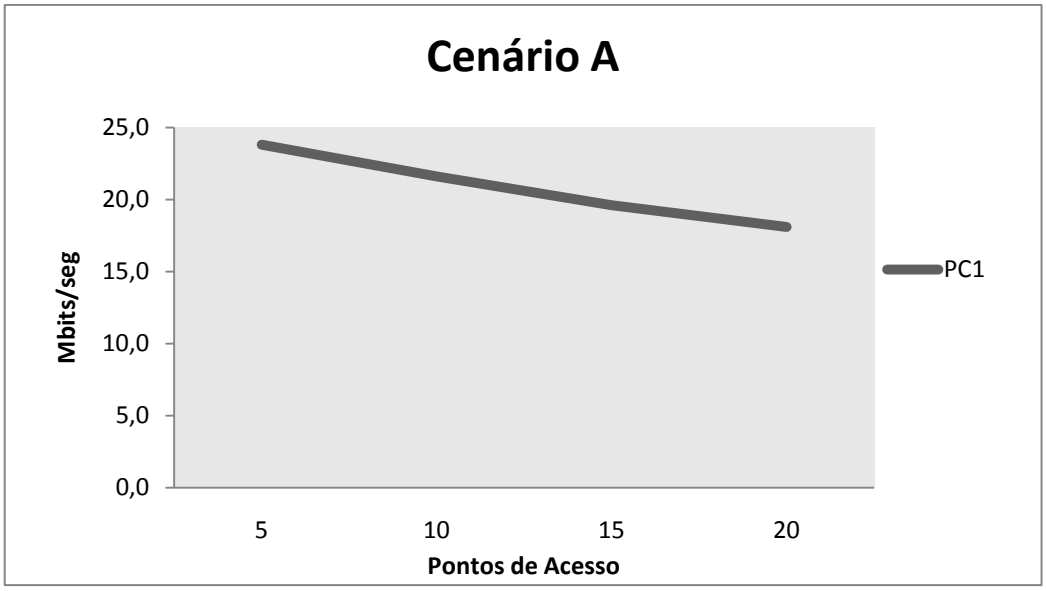

Figura 3.8 - Largura de banda referente à Configuração I, Cenário A.

A figura 3.8 mostra os valores do PC1 relativos à configuração I e ao cenário A. Torna-se visível que o aumento do número de pontos de acesso neste cenário influência directamente diminuindo a largura de banda disponível para o PC1.

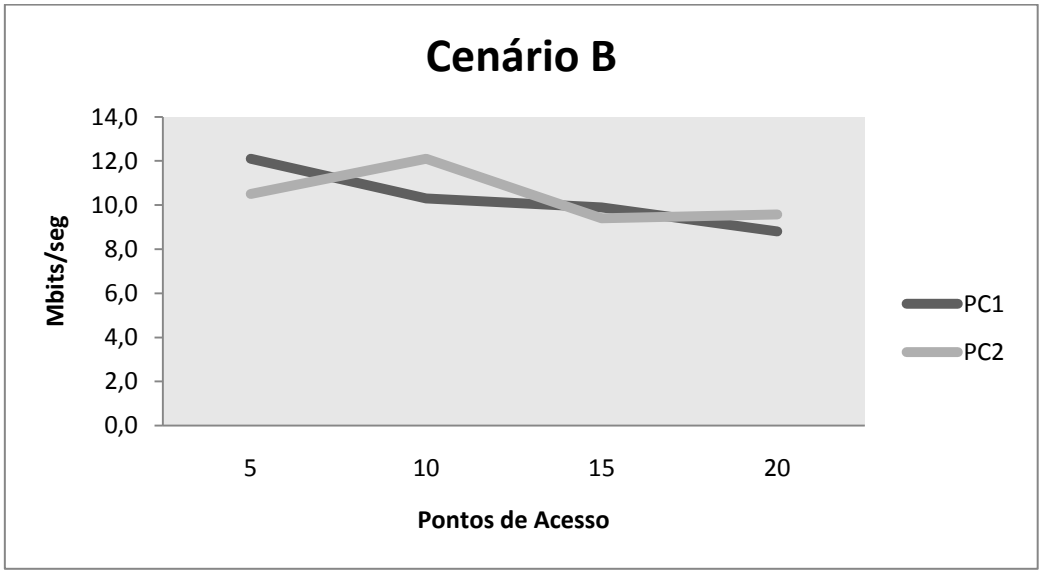

Figura 3.9 - Largura de banda referente à Configuração I, Cenário B.

A figura 3.9 representa os valores dos PC1 e 2 relativos à configuração I e ao cenário B em relação ao número de pontos de acesso. A diminuição da largura de banda

é visível ao comparar o valor inicial e o valor final, mas no entanto os valores intermédios demonstram algumas incoerências. Como não foi possível garantir distâncias iguais dos dois PCs clientes às estações emissoras, este poderá ser um facto para esta discordância de valores.

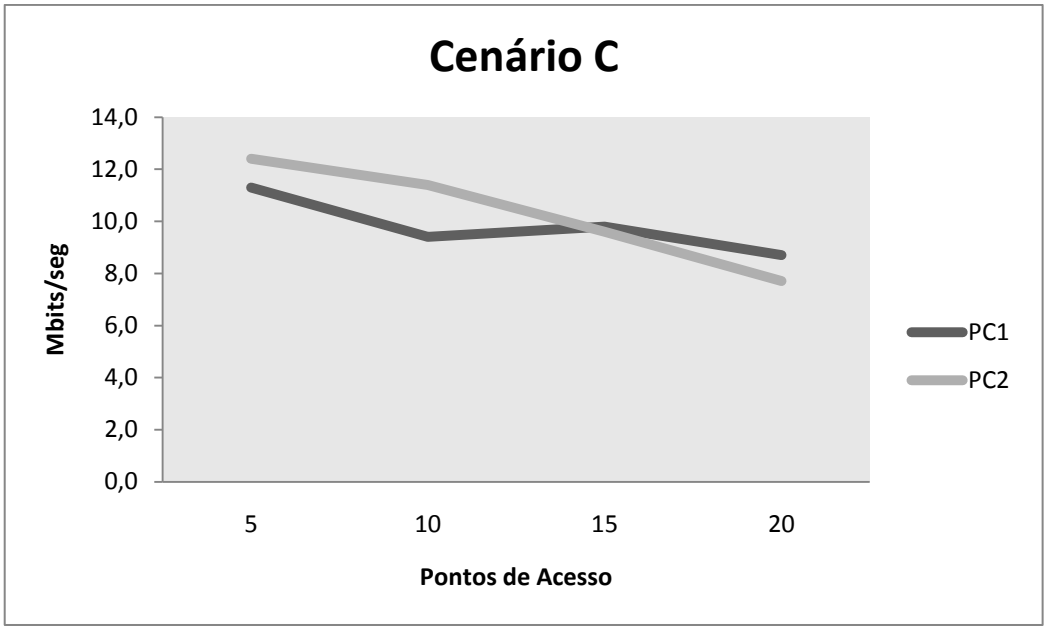

Figura 3.10 - Largura de banda referente à Configuração I, Cenário C.

O último gráfico, figura 3.10, representa os valores dos PC1 e 2 relativos à configuração I e ao cenário C. Assim como foi visível nos outros dois cenários para esta configuração, nesta figura verifica-se uma diminuição da largura de banda. Neste o PC1 ao contrário da figura anterior, é o que apresenta inconformidade ao obter uma subida de valor a meio da experiência. O factor explicado anteriormente relativo às distâncias entre as estações emissoras e os computadores clientes, poderá também para este caso ser a explicação do comportamento estranho apresentado na figura 3.10.

3.7.2 Throughtput vs Configuração II

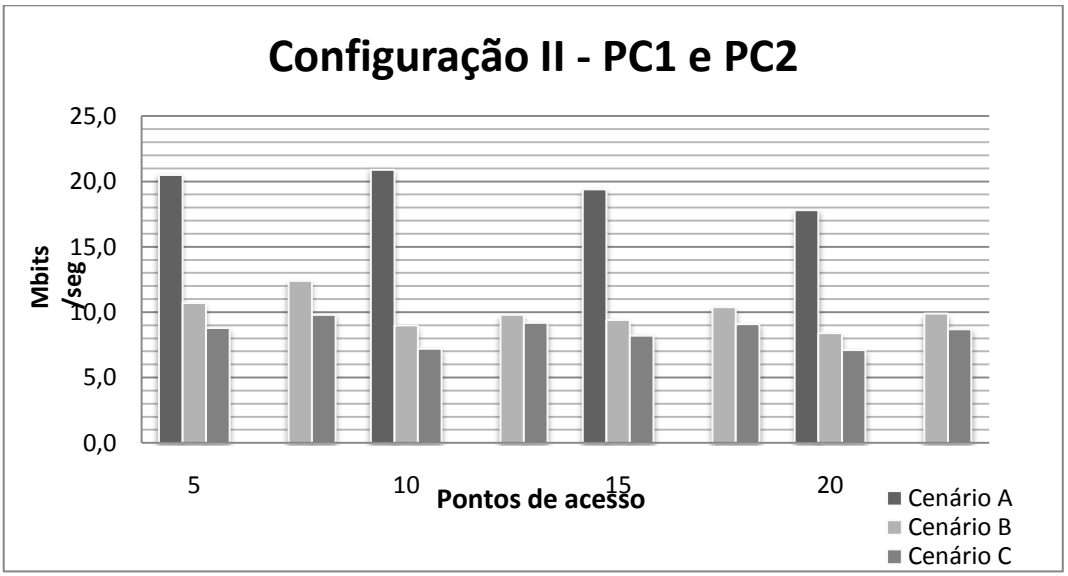

Figura 3.11 - Valores referentes à configuração II.

A figura 3.11 mostra os valores da experiência correspondentes á configuração II. Os valores dos vários cenários são apresentados em relação ao número crescente dos pontos de acesso do meio saturador. No gráfico da figura 3.11 verifica-se em termos gerais uma diminuição da largura de banda em todos os cenários para ambos os PCs (1 e 2). Algumas excepções estão também expostas, que serão comentadas aquando da apresentação detalhada de cada cenário.

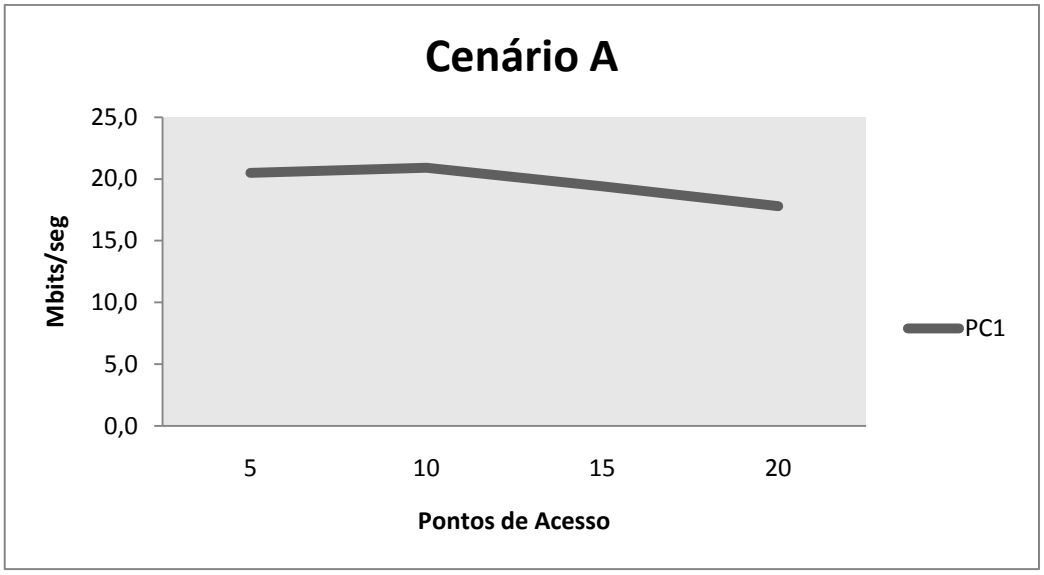

Figura 3.12 - Largura de banda referente à Configuração II, Cenário A.

A figura 3.12 é um gráfico onde é apresentado os valores da configuração II, cenário A, em relação ao aumento da densidade de pontos de acesso saturadores. Tendo em conta o valor inicial, obteve-se uma diminuição da largura de banda; no entanto um dos valores intermédios apresenta-se incoerente com a sequência dos valores obtidos.

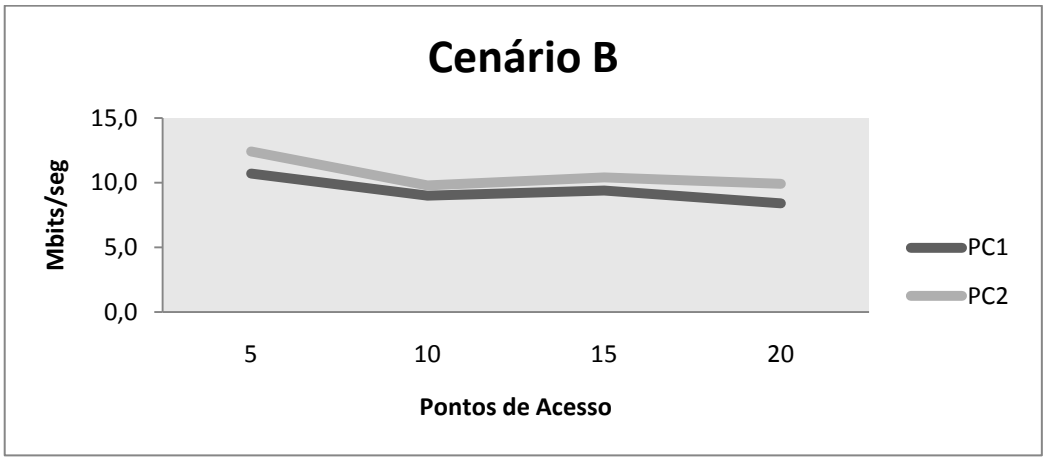

Figura 3.13 - Largura de banda referente à Configuração II, Cenário C.

Os valores referentes à mesma configuração, mas ao cenário B, são apresentados na figura 3.13. Nesta figura é possível observar um comportamento anómalo em ambos os clientes, primeiramente uma descida acentuada dos valores,

seguida de uma ligeira subida e finalizando com uma pequena descida. Mas, tendo em conta os valores iniciais e finais verifica-se uma diminuição da largura de banda.

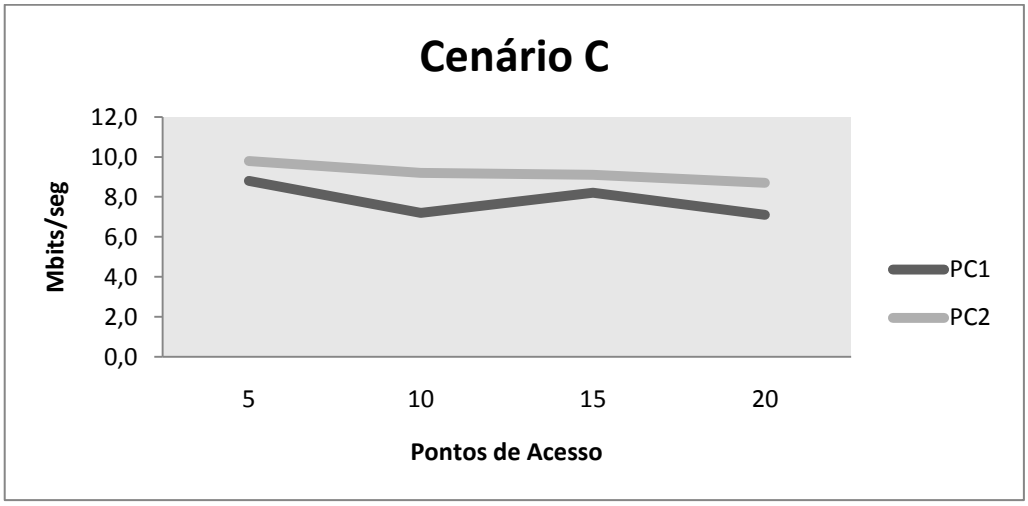

Figura 3.14 - Largura de banda referente à Configuração II, Cenário C.

A última figura desta configuração, figura 3.14, apresenta os valores da mesma configuração mas neste caso referente ao cenário C. No gráfico representado verifica-se uma ligeira descida da largura de banda tendo em conta os valores iniciais e os valores finais. No entanto, não se verificou um comportamento contínuo na diminuição da largura de banda disponível para o PC1. Este facto poderá estar relacionado com o meio onde se foi efectuado o estudo, visto que não foi possível garantir um meio onde não existissem outros pontos de acesso para além dos programadados para este estudo, assim poderá ter influenciado desta forma os resultados obtidos neste cenário.

3.7.3 Throughtput vs Configuração III

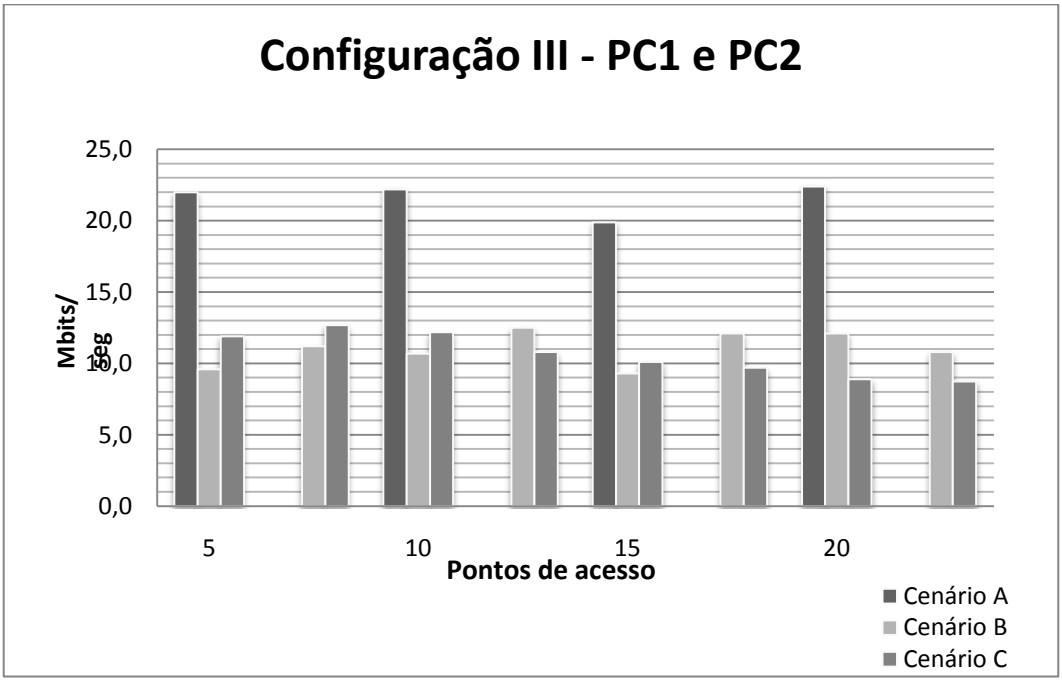

Figura 3.15 - Valores referentes à configuração III.

O gráfico representado na figura 3.15, relaciona a largura de banda dos clientes na terceira configuração com o aumento dos saturadores do meio. De uma forma geral, verifica-se uma diminuição da largura de banda disponível para os clientes. Para cada cenário desta configuração, os valores serão apresentados seguidamente.

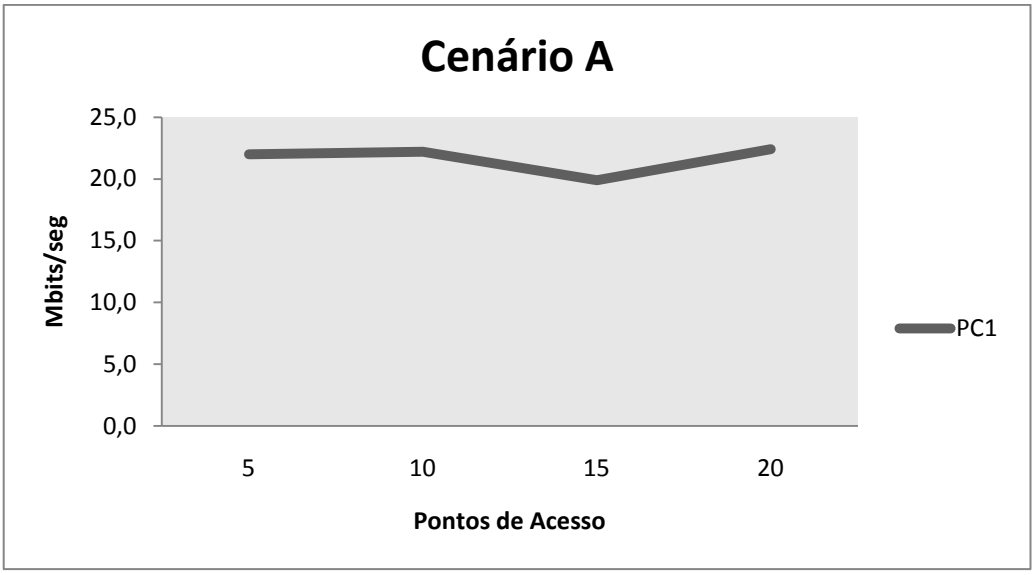

Figura 3.16 - Largura de banda referente à Configuração III, Cenário A.

A figura 3.16, representa a largura de banda do PC1 relativa ao aumento de saturadores do meio na configuração II, cenário A. Verifica-se um aumento do valor inicial para o valor final, apesar de nos valores intermédios se verificar uma diminuição da largura de banda disponível. Esta tendência não era esperada. A explicação para a tendência apresentada, pode estar relacionada com a interferência de outras redes existentes no meio, não pertecentes ao estudo, nos valores iniciais, fazendo com que estes apresentam-se valores menores que os valores finais.

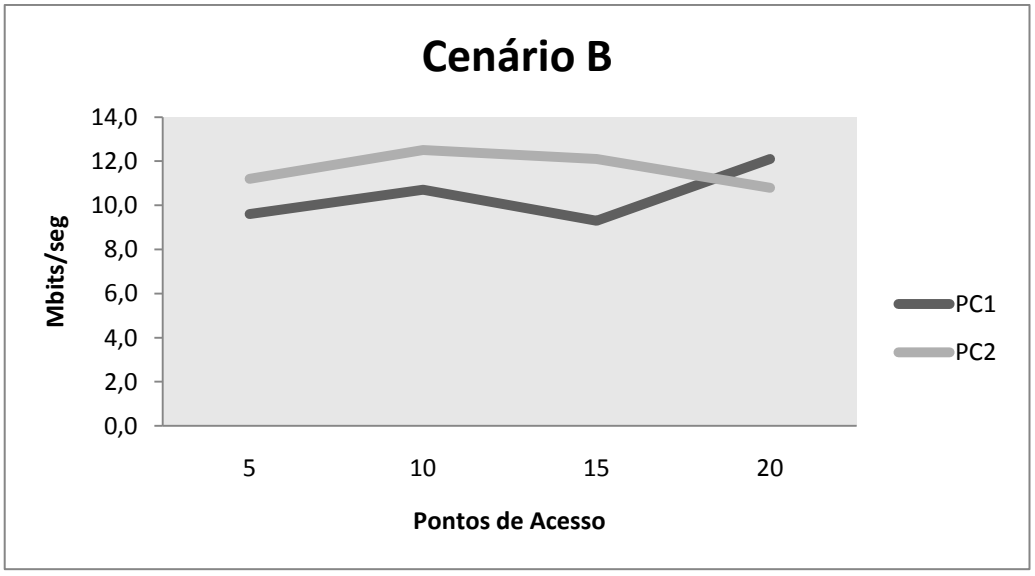

Figura 3.17 - Largura de banda referente à Configuração III, Cenário B.

O gráfico representado na figura 3.17 associa da mesma forma as larguras de banda com os pontos de acesso, mas neste caso relativamente ao cenário C da terceira configuração. É visível que os clientes tiveram comportamentos diferentes na parte final da experiência: o PC1 aumentou a largura de banda enquanto a largura do PC2 diminuiu. O facto do meio onde se efectou o estudo apresentar redes fora do âmbito do estudo poderá também neste caso ser a explicação para as tendências apresentadas.

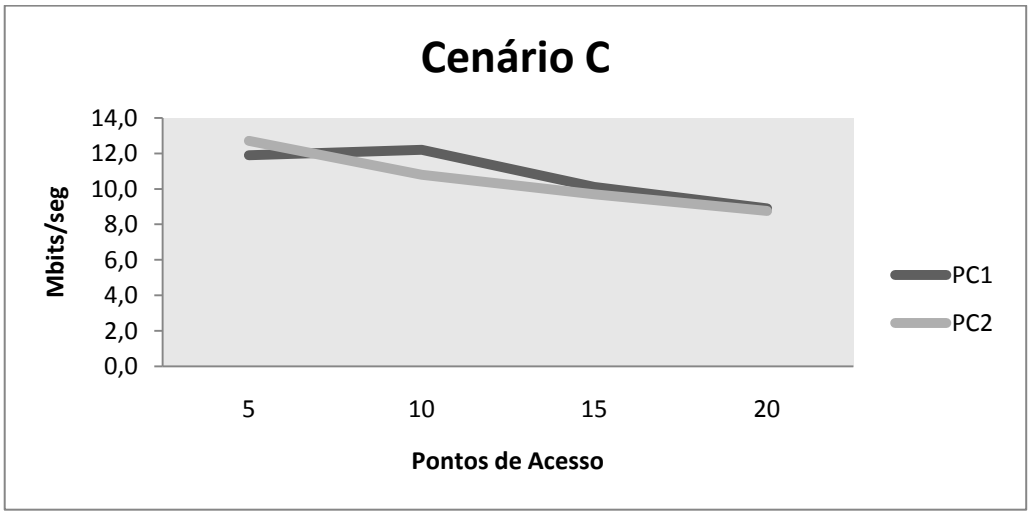

Figura 3.18 - Largura de banda referente à Configuração III, Cenário C.

O último gráfico da terceira simulação, está representado na figura 3.18, e diz respeito ao cenário C. É visível que em relação aos valores iniciais ambos os clientes apresentaram uma diminuição na largura de banda, no entanto nos valores intermédios verificam-se mais uma vez comportamentos opostos entre os clientes, com a mesma explicação possível, ausência de um meio exclusivo para o meio.

3.7.4 Throughtput vs Configuração IV

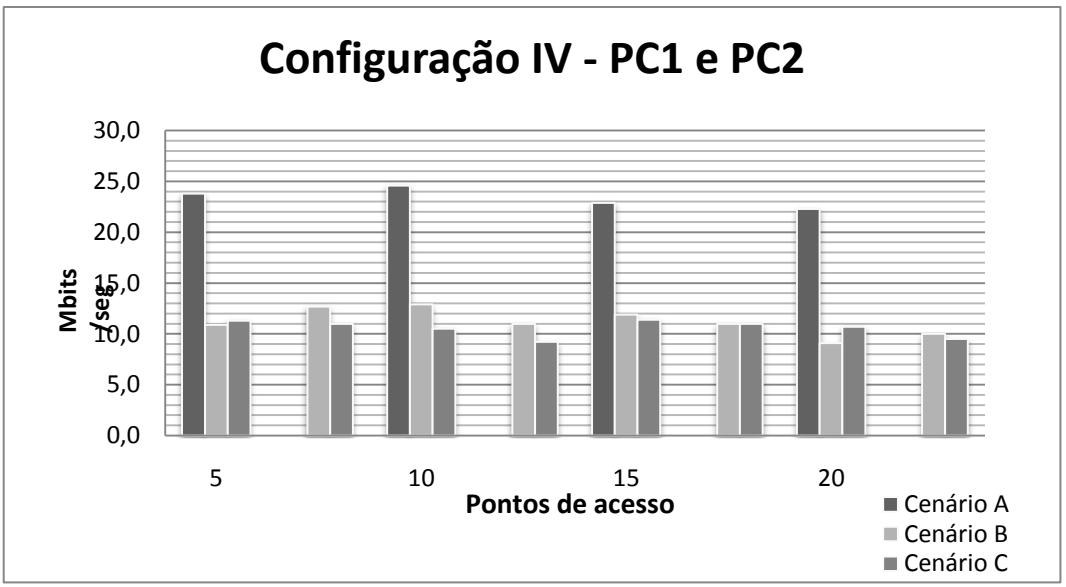

Figura 3.19 - Valores referentes à configuração IV.

Para finalizar, a figura 3.19 ilustra os valores da largura de banda referentes à quarta configuração. É visível num âmbito geral que os valores para a largura de banda diminuíram aquando a introdução de mais elementos saturadores no meio. Apresenta-se também um ligeiro aumento nos valores intermédio como facto anómalo.

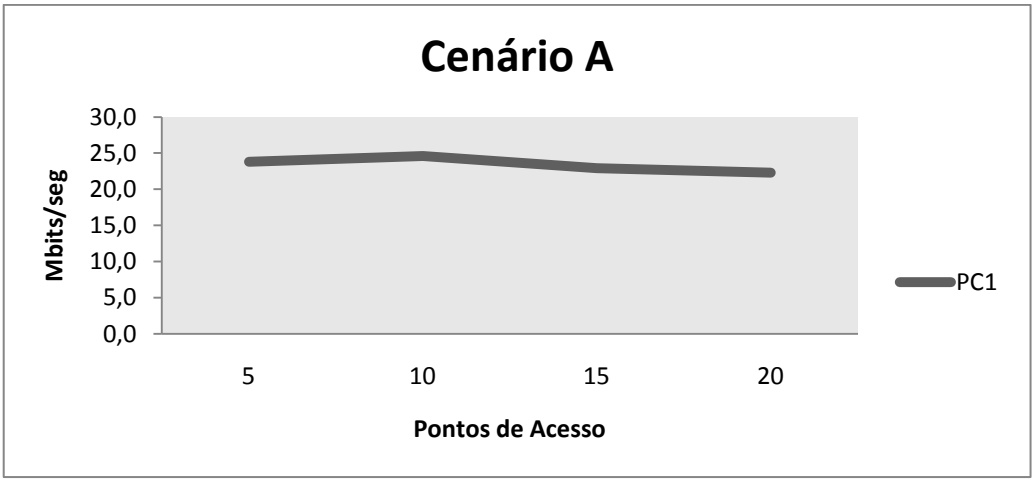

Figura 3.20 - Largura de banda referente à Configuração IV, Cenário A.

A figura 3.20 mostra um gráfico onde é demonstrada a relação entre a largura de banda do PC1 e o número de pontos de acesso introduzidos no meio envolvente à rede, no cenário A da quarta simulação. O comportamento verificado não foi regular mas, tendo em conta o valor inicial e final da simulação observa-se um ligeiro decréscimo na largura de banda disponível.

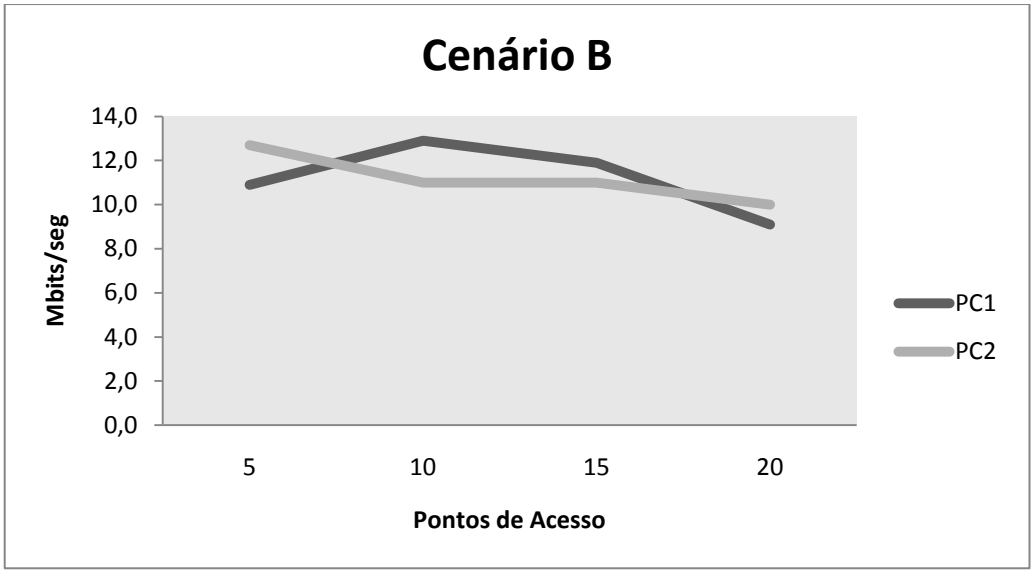

Figura 3.21 - Largura de banda referente à Configuração IV, Cenário B.

Da mesma forma que a figura 3.20, a figura 3.21, demonstra a mesma relação mas para o cenário B, onde intervêm dois clientes: PC1 e PC2. Em ambos os clientes registou-se uma diminuição da largura de banda disponível, no entanto os comportamentos durante a simulação dos clientes foi ligeiramente diferente: o cliente 1 subiu bastante, descendo na fase final da simulação; o cliente 2 decresceu nos momentos iniciais, mantendo-se seguidamente, e no final da simulação novamente apresentou-se com uma diminuição. Neste cenário, a falta de um meio usado apenas para este experiência é notório, pois a tendência é esperada, mas os valores intermédios, neste caso, não sofrem tanta interferência como os valores iniciais e finais, apresentando assim uma irregularidade dos valores.

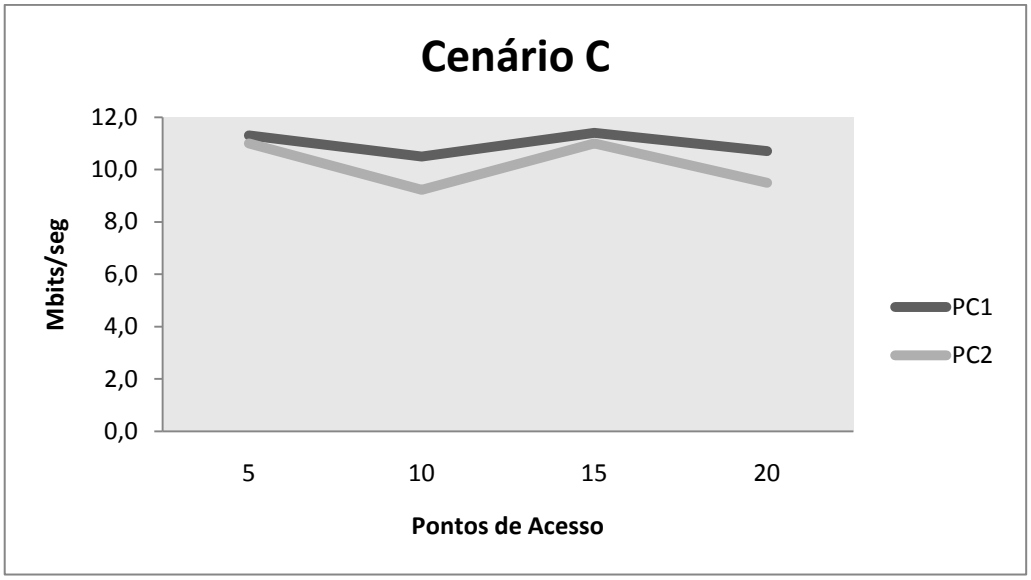

Figura 3.22 - Largura de banda referente à Configuração IV, Cenário C.

Para finalizar a quarta configuração, a figura 3.22 ilustra a relação entre a largura de banda e o crescente número de pontos de acesso como saturadores no meio envolvente aos clientes PC1 e PC2 no cenário C. Nesta figura, a linha de tendência apresenta uma inclinação mínima, o que nos demonstra que houve uma diminuição da largura de banda pouco acentuada, apesar de valores intermédios para o PC1 poderem violar com esta tendência. Mais uma vez, a interferência de outras redes não pertencentes ao estudo poderão ser a explicação directa para estes valores intermédios.

Após esta apresentação detalhada dos resultados em cada simulação, serão em seguida apresentados resultados globais de toda a experiência. Estes valores são importantes para posterior análise, reflexão e crítica aos estudos a realizar.

Valores médios (Mbits/seg)				
Cenário A				
	Configuração I	Configuração II	Configuração III	Configuração IV
PC1	20,8	19,7	21,6	23.4
Cenário B				
PC1	10,3	9,4	10,4	11,2
PC2	10,4	10,6	11,7	11,2
Cenário C				
PC1	9,8	7,8	10,8	11,0
PC2	10,3	9,2	10,5	10,2

Tabela 3.7 - Valores médios referentes a todas as configurações e cenários.

A tabela 3.7, mostra-nos os valores médios de todas as simulações. Com a análise dos valores apresentados, de uma forma generalizada é notável em todos cenários correspondentes a cada configuração uma diminuição da largura de banda disponível para a transmissão de dados. Tal como esperado, a saturação do meio apresenta influência directa nas redes *wireless* formadas nos vários cenários. Desta forma prova-se que num meio saturado partilhado por dois clientes ligados a pontos de acesso diferentes, existe uma diminuição acentuada na largura de banda disponível para transmissão de dados de ambos os clientes.

Outro aspecto importante a referir é o facto de, apesar de serem escolhidos canais não sobrepostos, o meio saturar-se também graças à densidade dos terminais. No entanto são visíveis diferenças notáveis nas várias configurações, em destaque para o cenário C. Da configuração II para a configuração III verifica-se um aumento da largura de banda dos clientes, no PC1, dos 7.8 Mbits/seg para os 10.8 Mbits/seg e no PC2, dos 9.2 Mbits/seg para os 10.5 Mbits/seg, em valores médios, o que nos demonstra que o uso de canais não sobrepostos melhora a largura de banda, apesar de se encontrar num meio saturado.

O aumento da largura de banda a um nível geral também se verificou na mudança de canais e na manutenção de SSID iguais (configurações I e IV), no PC1 de 9.8 Mbits/seg, na primeira configuração para os 11.0 Mbits/seg da última configuração. No PC2 verificou-se algo estranho, a largura de banda diminui em 0.1 Mbits/seg, ficando assim na última configuração nos 10.2 Mbits/seg Este facto, estará relacionado com o número de amostras obtidas para estes testes; com um número de amostras maior, esta diminuição provavelmente não se apresentaria.

Foi ainda possível verificar que, o facto de os terminais do meio saturador usarem SSID diferentes em canais diferentes, é a melhor situação encontrada. Provoca uma menor diminuição na largura de banda disponível para os clientes do AP1 e do AP2. Com SSID iguais em canais iguais, obteve-se valores de 9.8 Mbits/seg e 10.3 Mbits/seg para o PC1 e PC2 respectivamente, enquanto com SSID diferentes em canais diferentes, os valores são de 10.8 Mbits/seg e 10.5 Mbits/seg respectivamente aos PCs 1 e 2.

Neste estudo foram usadas placas *wireless* idênticas para que o estudo fosse o mais imparcial possível. No entanto, não foi possível garantir distâncias iguais do PC1 para o AP1, e do PC2 para o AP2. Este facto é visível na maior parte dos gráficos onde se verifica que o PC2 apresenta normalmente maior largura de banda que o PC1. Estas pequenas distâncias, acrescido do número baixo de medidas tiradas para este estudo (três medidas, obtendo uma média), fizeram com que algumas incoerências aparecessem durante as simulações. No entanto apesar destas imprevisões, tendo em conta os valores iniciais e os valores finais em cada uma das simulações foi possível obter conclusões credíveis.

Num âmbito mais geral é possível concluir que os elementos saturadores levaram à diminuição do desempenho das redes estabelecidas entre o PC1 e AP1 e o PC2 e AP2. Isto é, a largura de banda diminui, em alguns casos drasticamente, devido ao ruído e à sobreposição de canais existente no meio envolvente às redes em que os clientes operavam.

3.7.5 Throughtput vs Sinal

Através do NetStumbler [**44**] no computador analisador, foi possível obter os valores médios de SNR dos pontos de acesso saturadores do meio envolvente. Mas tendo em conta que a placa de rede sem fios do analisador usa o *Network Driver Interface Specification* (NDIS) *driver,* os valores de ruído têm um valor fixo de -100, o que significa que neste caso o *Signal to Noise Radio (***SNR)** medido é igual ao **sinal.** O seguinte gráfico refere-se apenas ao cenário C em todas as configurações, visto este cenário ser o mais importante para o estudo (pois permite verificar o comportamento de duas redes distintas impostas ao mesmo meio envolvente).

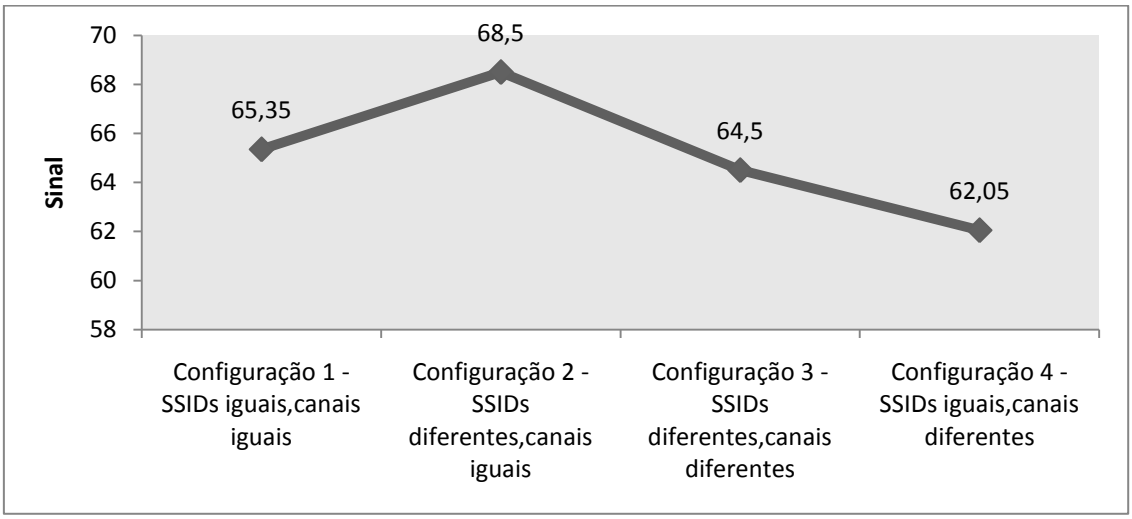

Figura 3.23 - Valores de sinais de potência de cada configuração.

A figura 3.23 relaciona o sinal com cada uma das configurações do estudo no estado final, onde constam vinte pontos de acesso saturadores do meio. Pelo gráfico da figura é notável que na configuração II se registou o pico desta medição. Este valor demonstra que o facto de se concentrarem vários pontos de acesso no mesmo canal mas com SSIDs diferentes, leve a que em média os pontos de acesso têm que potenciar um maior sinal do que se esta densidade de terminais estivesse espalhada pelo espectro em canais diferentes e em SSIDs também diferentes, caso da configuração III.

Os dois maiores valores são assim registados aquando da utilização de um único canal para todos os pontos de acesso, tendo ou não SSIDs iguais. Desta forma os dois

maiores valores registados coincidem aquando se obtém menores larguras de bandas (tabela 3.3), o que permite concluir que no primeiro caso, configuração I, com SSIDs iguais, os pontos de acesso, apesar de apresentarem um bom sinal, o ruído por eles introduzidos no mesmo canal era intenso o que provocava uma diminuição acentuada na largura de banda disponível para os clientes. Mudando os SSIDs, mas mantendo apenas um canal de operação, fez com que os sinais dos pontos de acesso aumentassem, mas por consequência o ruído introduzido pelos mesmos no meio ambiente fosse também maior, obtendo assim os piores valores do estudo da largura de banda disponível para os clientes.

Os dois menores valores referem-se assim às configurações III e IV. Estas configurações têm em comum o uso de canais diferentes. No entanto com SSIDs iguais (IV) obteve-se um pior registo de sinal, correspondendo aos melhores valores de largura de banda disponível para os clientes. Com isto se conclui que como os pontos de acesso enviam pouco sinal para o meio, este interfere menos com as redes que nesse ambiente operam. Para a configuração III obteve-se o segundo pior registo, que coincide com o segundo melhor registo em apreciação global da largura de banda disponível para o cliente. Conclui-se portanto que o facto de em canais diferentes existirem redes diferentes, faz com que os pontos de acesso tenham um pior sinal. No entanto também não apresenta grande ruído, visto que esta configuração apresenta valores aceitáveis para a largura de banda.

De uma forma generalizada se pode concluir que o facto dos pontos de acesso apresentarem menos sinal não significa que seja introduzido mais ruído no meio ambiente, como é comprovado pela largura de banda disponível em cada uma das simulações. Por isso, neste caso, a largura de banda cresce inversamente proporcional ao sinal dos pontos de acesso.

3.7.6 Throughtput vs Espectro 2.4GHz

Graças ao programa Chanalyzer [**45**] a correr no analisador, foi possível analisar a amplitude de todos os canais no espectro dos 2.4GHz. Desta forma podemos comparar a

amplitude inicial e final dos canais em cada uma das configurações para o cenário C e relacionar com largura de banda disponível para os clientes.

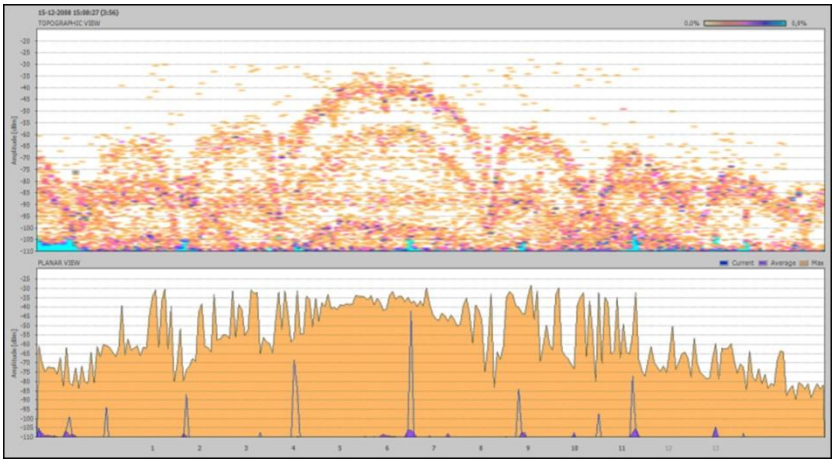

Figura 3.24 - Estado inicial do espectro na configuração I.

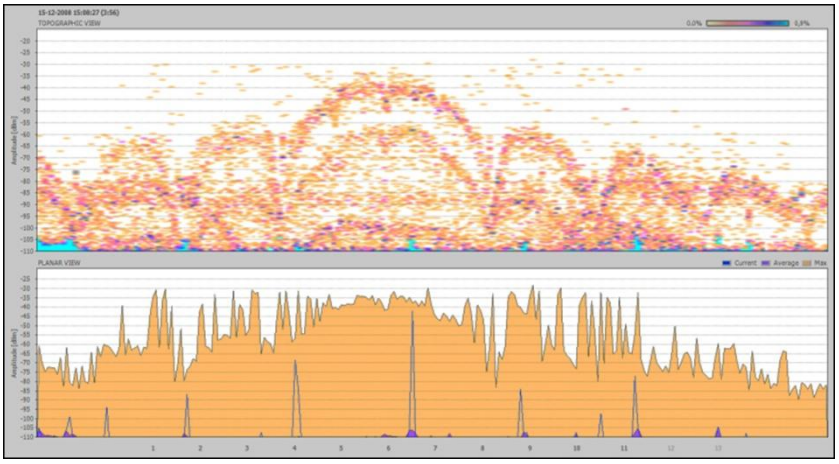

Figura 3.25 - Estado final do espectro na configuração I.

As figuras 3.24 e 3.25, representam respectivamente, o estado inicial e o estado final do espectro durante as simulações referentes à configuração I. É possível verificar um aumento da densidade dos pontos, o que demonstra a saturação do meio envolvente.

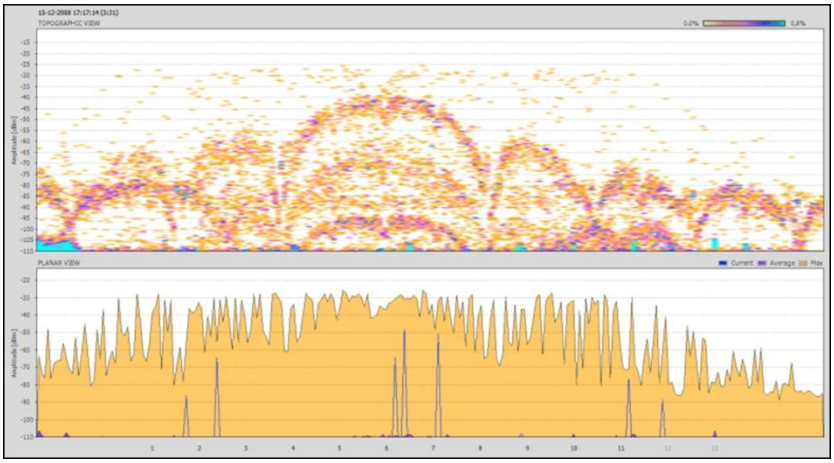

Figura 3.26 - Estado inicial do espectro na configuração II.

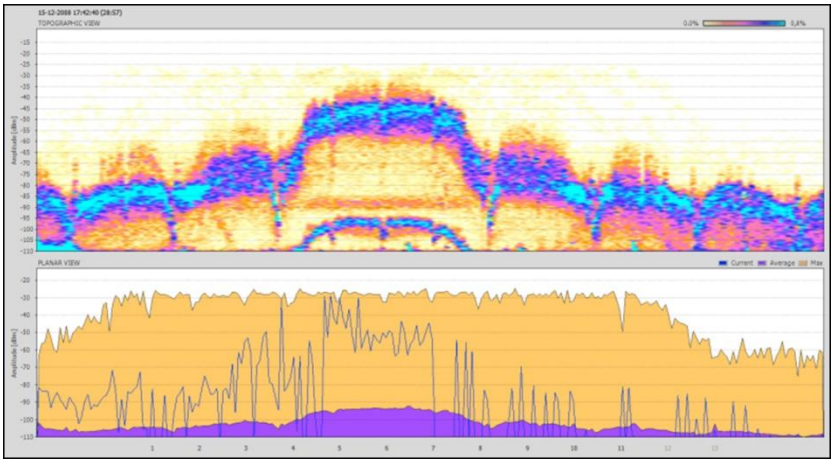

Figura 3.27 - Estado final do espectro na configuração II.

As figuras 3.26 e 3.27, ilustram os momentos iniciais e finais das simulações referentes à configuração II. É possível notar que mantendo os pontos de acesso todos no mesmo canal, verifica-se uma densidade provocada por essa sobrecarga.

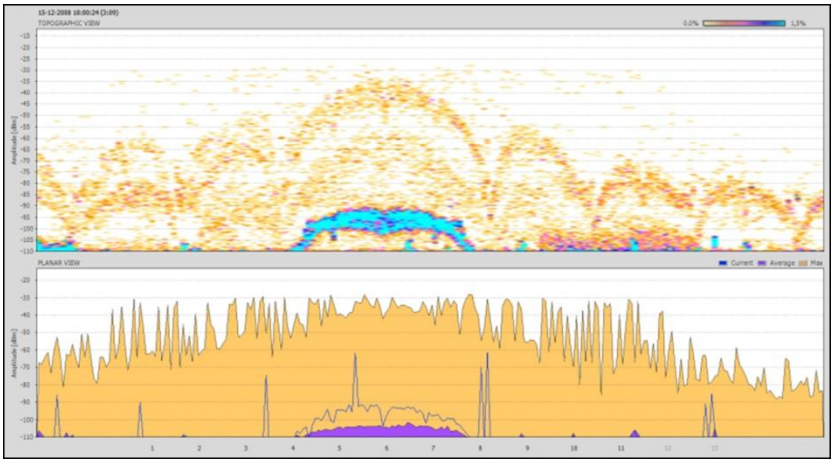

Figura 3.28 - Estado inicial do espectro na configuração III.

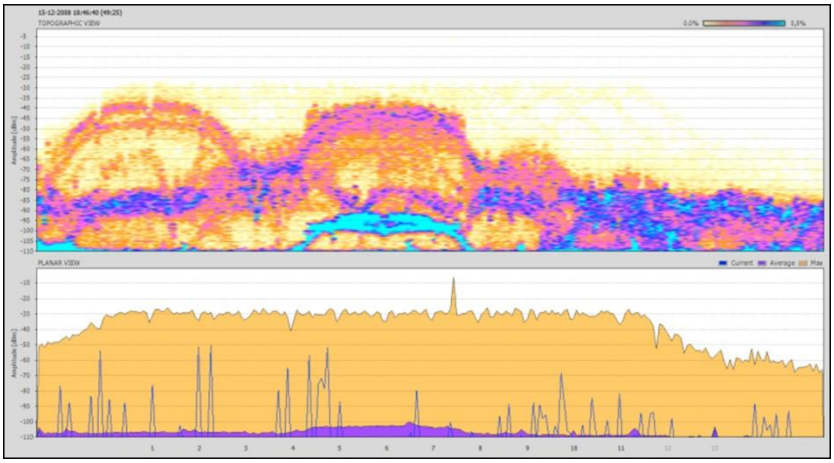

Figura 3.29 - Estado final do espectro na configuração III.

As imagens ilustradas pelas figuras 3.28 e 3.29, dizem respeito aos momentos iniciais e finais das simulações da configuração III. É notável que o espectro não se encontra em sobrecarga apenas em um canal, mas em vários, obtendo assim uma sobrecarga generalizada e não uma sobrecarga apenas centralizada.

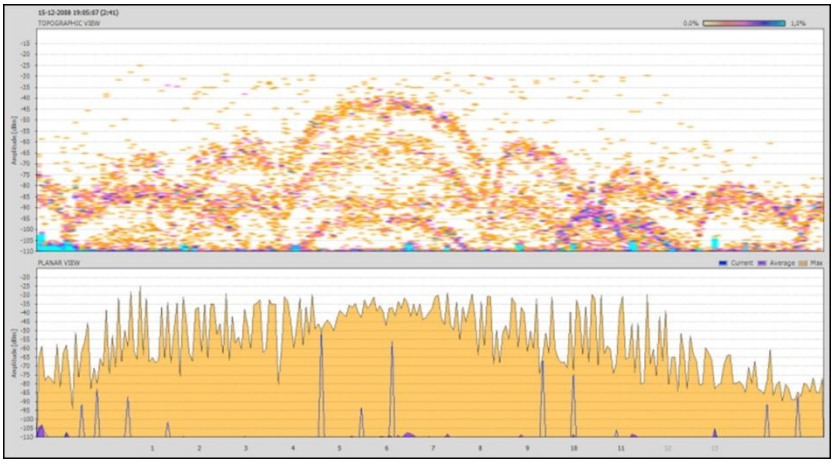

Figura 3.30 - Estado inicial do espectro na configuração IV.

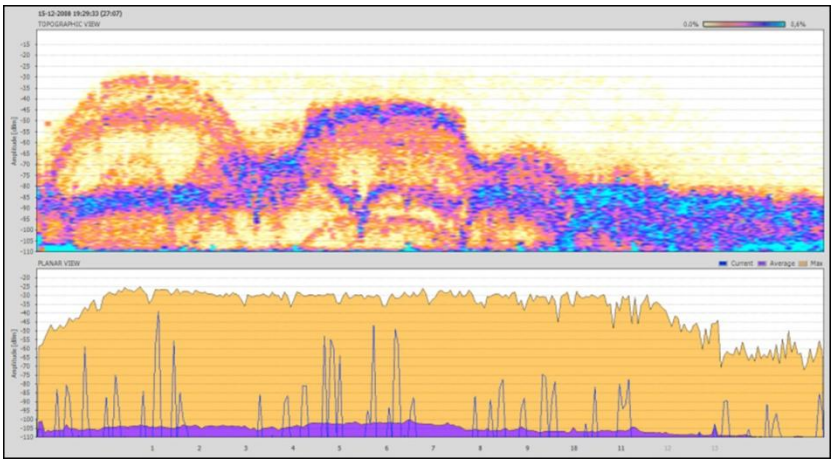

Figura 3.31 - Estado final do espectro na configuração IV.

Por último, as figuras 3.30 e 3.31, referem-se aos estados iniciais e finais das simulações referentes à configuração IV. Nestas imagens é possível observar uma sobrecarga ainda maior que na configuração III. Apesar de ser uma sobrecarga a nível global, o facto de se usarem SSIDs iguais influenciou directamente de forma a existir uma sobrecarga pesada em todo o espectro.

É visível que em todas as configurações, no início o espectro apresenta-se com algum sinal rádio, mas sem intensidade e não cobrindo toda a gama dos 2.4 GHz do espectro, devido ao facto de no local em estudo existirem outras redes sem ser as referidas no estudo. Em contraste, nas vistas finais depara-se com um ambiente saturado, onde a gama se encontra completamente superlotada. O meio em todos os estados finais encontra-se obviamente saturado, como era esperado no estudo, o que, como já foi

concluído, reduz (em alguns casos muito acentuadamente) a largura de banda disponível para transmissão.

Nas configurações I e II ao contrário das duas seguintes, pode-se observar que o espectro se encontra concentrado no canal 6 do espectro, onde se pode observar uma grande densidade com a curvatura própria de um sinal rádio de 802.11b/g. Esta curvatura deve-se ao facto como sinal é modulado. Nas configurações III e IV, o espectro apresenta duas curvaturas correspondentes ao sinal 802.11b/g, neste caso referentes ao canal 1 e 6, como era desejado. O espectro apresenta-se neste caso saturado, mas de uma forma mais uniforme. Assim foi possível concluir o estudo verificando de uma forma visível a saturação no espectro dos 2.4 GHz.

3.8 Sumário / Conclusões

Neste capítulo foi apresentado um processo onde se pretendia avaliar as consequências para a Universidade de Aveiro aquando da saturação do meio. Com este estudo, foi possível provar que a saturação do meio envolvente provoca aos pontos de acesso em redes infra-estruturadas interferências nas ondas rádio, conseguindo diminuir, em alguns casos drasticamente, a largura de banda disponível para as estações que comunicam com o emissor. Desta forma, ficou comprovado o que foi apresentado teoricamente no capítulo 2.

Este capítulo apresenta-se como um estudo complementar ao próximo capítulo, capítulo fulcral desta dissertação, onde será estudada a topologia dos pontos de acesso 802.11 que se encontram pela cidade de Aveiro.

Capítulo 4

Caracterização de redes sem fios não planeadas

4.1 Introdução

A crescente necessidade actual das pessoas acederem à informação levou a várias cidades optar hoje por oferecer serviços de internet aos seus habitantes, através de redes organizadas estruturalmente. Desta forma é possível numa cidade ter acesso à internet numa dada área coberta, eventualmente grátis. A cidade de Aveiro acompanha a evolução tecnológica e, este cenário pode ser interessante para o futuro.

O primeiro passo para a construção de uma rede deste tipo é a análise dos APs em Aveiro. Assim, esta cidade será a cidade alvo de um estudo aprofundado da estrutura actual de todas as redes privadas e públicas existentes. Tendo em conta a análise feita, é possível obter uns valores estatísticos e estimar outros. Com este estudo, pretende-se essencialmente, conseguir obter um modelo ideal para a análise desta nova rede. O presente capítulo apresenta o estudo acima referido, explicando todo o seu conteúdo pormenorizada, e, por fim, descreve as características da topologia encontrada.

4.2 Descrição do trabalho

Após a apresentação no capítulo dois de várias redes organizadas em várias partes do mundo, pretende-se estudar a cidade de Aveiro, de forma a permitir a estruturação de uma rede que

ofereça à cidade um serviço de internet. Assim o objectivo fulcral é estudar a viabilização de uma infra-estrutura para uma rede sem fios em malha não planeada na cidade de Aveiro.

Este estudo apresenta uma sequência de tarefas à qual não era possível alterar a sua ordem, mas que pretendia alcançar o objectivo primal do estudo. Em primeiro lugar, tendo em vista a projecção de uma infra-estrutura desta dimensão na cidade, era necessário fazer um mapeamento da cidade de forma a contabilizar e localizar todos os equipamentos existentes que suportassem a norma IEEE 802.11, exposta no capítulo anterior. Esta recolha informatizada foi feita através de um *warbiking* pela cidade. Para tal, foi usado uma bicicleta (ver figura 4.1), equipada com um portátil, uma placa de redes sem fios específica, uma antena específica e dois dispositivos de localização geográfica (GPS). Procedeu-se assim à recolha dos dados da cidade inteira.

Figura 4.1 - Bicicleta usada para o *warbiking*

Em segundo lugar, com a recolha informática realizada, pretendia-se em ambiente de simulação obter resultados indicativos de uma rede comunitária municipal. Para transpor a topologia encontrada anteriormente num ambiente de simulação, foi desenvolvido código em ambiente Linux para que estes dados fossem valores de entrada válidos nos simuladores. Os resultados obtidos referiam-se essencialmente à topologia que a cidade poderia apresentar. Os factores que influenciassem questões de desempenho da rede eram o ponto fulcral no estudo, para desta forma projectar da melhor forma uma rede sem fios em malha na cidade de Aveiro. Para obter estes resultados era necessário um processo de transformação dos dados, um processo intermédio no estudo, estruturando os dados.

Em terceira e último lugar: a simulação. Esta, foi de facto, o ponto dominante neste estudo, pois era o que permitia caracterizar a topologia da rede sem fios não planeada encontrada na

cidade de Aveiro. Nesta fase o objectivo é verificar, determinar e analisar até que ponto a topologia encontrada permitia a estruturação de uma rede sem fios em malha na cidade de Aveiro. No entanto, simular uma cidade com muitos pontos como foram encontrados na cidade de Aveiro não foi tarefa fácil. Ao longo do trabalho foi necessário adoptar novas estratégias, recorrendo a diferentes simuladores. Por fim, dada a dificuldade em simular cenários de grande dimensão nos programas existentes para simulação, reestruturou-se esta fase com recurso à construção de um grafo. Usando um dado alcance estabelecido como limite, transformou-se a cidade mapeada num grafo, de forma a obter todos os resultados estatísticos possíveis. Com base nesses resultados foi caracterizada a possível rede não planeada e assim concluído o estudo.

4.3 Cenários Geográficos

Sendo proposta a cidade de Aveiro como a cidade alvo deste estudo, o cenário é exposto no mapa da cidade de Aveiro (figura 4.2 e figura 4.3). Aveiro é um concelho com uma área total cerca de 200 Km^2 e com uma densidade populacional de 368,4 hab/ Km^2 [**47**]. Tendo em conta o tamanho da cidade, e o facto de se pretender apenas estudar o centro de Aveiro redimensionou-se o estudo para dimensões mais execuíveis. Um estudo ao concelho apresentar-se-ia como um desafio enorme, que para o pretendido apenas se tornaria em algo exaustivo.

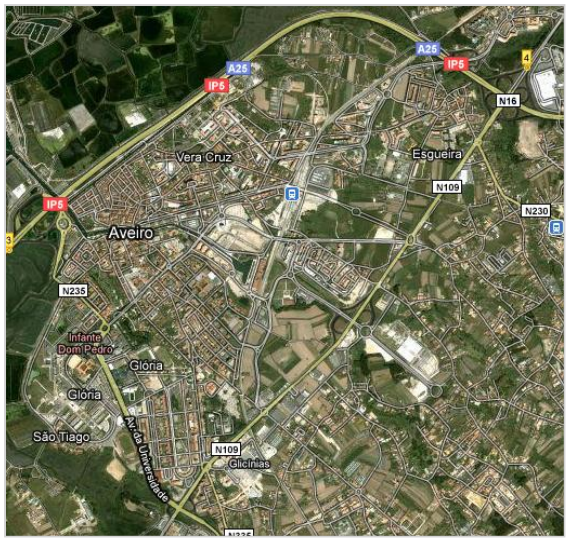

Figura 4.2 - Mapa da cidade de Aveiro via satélite.

Figura 4.3 - Esquema do mapa da cidade de Aveiro.

As figuras 4.2 e 4.3 ilustram o cenário deste estudo. Pela imagem da figura 4.3, é possível focar a área a que o estudo foi limitado. Para este limite foi tido em conta a autoestrada A25, a estrada nacional 109 e a Avenida da Universidade, contando com a zona de São Tiago e parte da zona da Glória, zona que contêm a universidade.

Posto isto, torna-se necessário realçar alguns dados estatísticos da área em estudo. Os dados estatísticos obtidos foram calculados a partir dos dados disponíveis em [**47**]. A área de estudo abrange essencialmente três freguesias: Glória, Vera Cruz e Esgueira. Os dados estatísticos destas freguesias são apresentados na tabela 4.1.

Freguesias	**Área total (Km^2)**	**Densidade Populacional (Hab/Km^2)**	**Área abrangida pelo estudo (% área total - Km^2)**	**Área habitada abrangida pelo estudo (% área total - Hab)**
Glória	6.87	1 445.00	55 % - 3.7785	100% - 9 927
Vera Cruz	38.48	224.87	15 % - 5.7720	100% - 8 650
Esgueira	17.76	690.00	15 % - 2.6640	30% - 3 676
	Total	716.67 ≈ 716	12.2145 ≈ 12 Km^2	22 253 ≈ 22 300 Hab

Tabela 4.1 - Dados estatísticos referentes à cidade de Aveiro **[**47**]**.

A tabela 4.1 mostra os dados estatísticos das freguesias envolvidas na área de estudo. Tendo em conta o mapa da cidade, aproximadamente obteve-se um valor estimado para a área total abrangida pelo estudo, cerca de 12 Km^2. Este valor em termos geométricos, e em comparação com a forma da área de estudo, refere-se a um rectângulo de 4 Km por 3 Km, obtendo assim os 12 Km^2. Sendo assim este valor significa cerca de 6% do concelho de Aveiro, o que nos mostra a limitação que foi necessário efectuar. Mesmo assim corresponde a mais de 40% da população

Verificando, que mesmo com o cenário reduzido, a área que abrangia o estudo era ainda considerável, foi necessário dividir a área de estudo por zonas. Esta divisão serviu para uma simplificação de todo o processo de recolha da localização dos pontos.

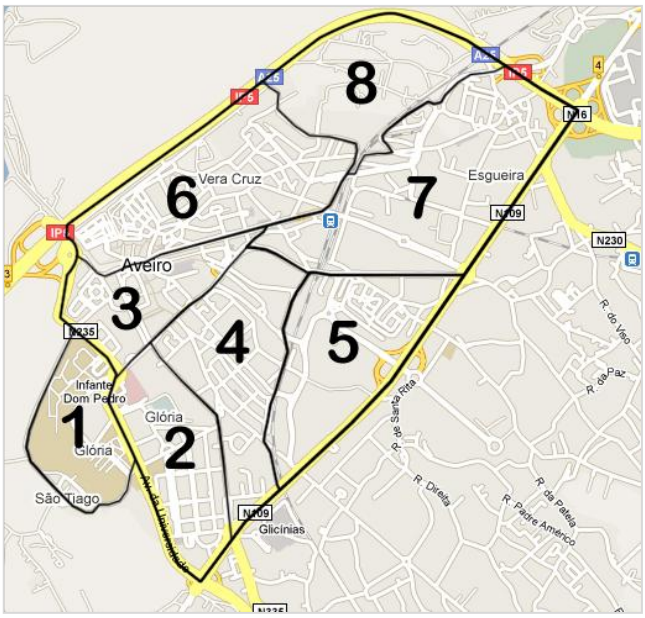

Figura 4.4 - Divisão da área em estudo em oito zonas.

Na figura 4.4 é apresentada a divisão da cidade em zonas. O objectivo desta divisão era, facilitar a recolha de dados: era possível ter uma melhor percepção do percurso efectuado e do percurso restante para concluir a recolha numa dada zona. As zonas foram delimitadas tendo em conta estradas principais e locais onde a sua passagem fosse facilmente identificada. Esta divisão foi uma mais-valia, pois foi necessário repetir a recolha devido a ocasionais problemas técnicos, evitando assim a necessidade de uma nova recolha da área total em estudo.

Após tratada toda a informação relativa ao cenário, foi necessário implementar um método que se aplicasse a todas as zonas, e que se atingisse o objectivo final: caracterização de uma rede não planeada implementável na cidade.

4.4 Metodologia

Como dito, o estudo foi dividido em três processos importantes: localização dos pontos 802.11 e consequente recolha dos dados, conversão dos dados obtidos em dados de entrada válidos para o simulador e finalmente, extracção de dados do simulador para posterior caracterização da rede. Estes processos encontram-se pela ordem de implementação, pois este método apresenta-se como um acumular de informação até chegar ao veredicto final. Os três processos serão descritos seguidamente.

4.4.1 Aquisição

A localização dos pontos e recolha dos dados, foi feita através de w*arbiking*. Este método pretende, fazer os percursos das zonas apresentadas anteriormente, e através do uso de uma bicicleta equipada com o equipamento adequado (portátil, placa e antena de recepção de sinal *wireless* e GPS) conseguir guardar a informação aquando a passagem pelos diferentes locais. Existem vários conceitos deste tipo onde o que muda é o meio de deslocação: *wardriving* (utiliza o carro ou o autocarro), *warwalking* (é feito a pé) e ainda *wartramming* (onde o comboio é meio utilizado como meio de deslocação).

É pouco comum usar o *warbiking* neste tipo de recolha de dados, tendo em conta a área da cidade e distância total de todos os percursos. No entanto, é um método intermédio entre o *wardriving* e o *warwalking*, excluindo desde logo o *wartramming*, visto o que comboio apenas tem linha numa parte mínima da cidade. Segundo vários estudo, o erro de posicionamento dos pontos localizados é maior no *wardriving* do que no *warwalking*; o *warwalking* devido à lenta velocidade com que efectua a recolha, diminui o número de pontos não localizados, isto é, aumenta o número de pontos de acesso 802.11 localizados, pontos que com *wardriving* não seriam perceptíveis ao aparelho receptor [**48**] [**49**]. Outra possibilidade seria fazer um agregado dos dois métodos, neste caso, *wardriving* com um *warwalking*. Apresenta-se como boa alternativa, mas tendo em conta a relação tempo/precisão, um *warbiking* foi a escolha de melhor compromisso.

Tendo em conta a área da cidade e a superfície planar que apresenta, encontrou-se a solução ideal: *warbiking*, pois com velocidades intermédias entre os dois conceitos anteriormente comparados, não se tornaria tão exaustivo em relação a um *warwalking*, nem tão ineficaz que um *wardriving.*

Apesar deste cuidado, existe um erro à qual não se consegue evitar em nenhum dos métodos, a altitude de um dado ponto de acesso. Tendo em conta que existem pontos que se localizam em edifícios, e, que podem encontrar-se em diferentes pisos do mesmo prédio, estes durante a passagem não são diferenciados. Do ponto de vista 3D, num modelo cartesiano, os pontos "encontram-se" à mesma altura: todos os pontos de acessos são posicionados teoricamente no solo.

Seguindo o método escolhido e anteriormente resumido, fizeram-se duas passagens por dia, na mesma zona: primeira passagem das 10h às 12h, segunda passagem das 18h as 20h durante os dias 17 a 30 de Julho. As duas passagens têm como

objectivo conseguir obter um número de pontos de acesso 802.11 mais próximo da realidade; assim ao fazer duas passagens evita-se erros de percurso ou erros de captura de sinal pois é feito um cruzamento das duas passagens de forma a obter valores da recolha não sobrepostos. A ordem da recolha e localização dos pontos que foi utilizada, foi a ordem da distribuição da cidade por zonas, isto é, de forma crescente, começou-se pela zona um terminando na zona oito.

Os percursos das passagens em cada zona, foram estudados tendo em conta a estrutura da cidade, o sentido das ruas e sua dimensão. Isto permitiu que com ajuda do GPS, fosse possível fazer a passagem em todas as zonas, por todas as ruas, tentando não esquecer nenhuma parte do percurso.

Todo o equipamento utilizado para a recolha foi devidamente testado e configurado para o melhor funcionamento. Desta forma serão devidamente todas as especificações técnicas do equipamento necessário para a localização na tabela 4.2.

Portátil	
Marca / Modelo	Compal / HEL-80
Processador	Processador Intel Core 2 Duo T7200
Memória	2048Mb RAM DDR2
Bateria	9 Células (duração ~3h30m)
Placa *wireless*	Ubiquiti SRX 100mW [**50**]
Antena	Omnidireccional 9dBi
S.O.	Linux Ubuntu 9.04 [**42**]
Software	Kismet R1-06-2009 [**51**] (configuração: Anexo I)
GPS	
Receptor para o Kismet	Wintec WBT-201 serial/usb [**52**]
Receptor para orientação do percurso	Garmin Nuvi 300 [**53**]

Tabela 4.2 - Especificações técnicas do equipamento utilizado no *warbiking*

A escolha da placa e da antena foi cuidadosamente ponderada. Essencialmente, a sensibilidade neste tipo de equipamentos é o factor crucial para uma boa recolha de dados. A placa e a antena escolhida respondiam a este requesito.

O GPS, por sua vez, como todos os aparelhos de medição, apresenta um erro na sua medição. Para diminuir este erro é necessário ter em conta a velocidade, a localização de forma a evitar o *Urban Canyon Effect* [**54**] e a meteorologia. Assim, fez-se um *warbiking* moderado em velocidade, durante o mês de Julho, mês de verão, sem problemas de chuva ou nuvens, e pela estrada tendo sempre em atenção alguma falha de recepção de sinal dos

satélites. Para melhorar a precisão dos dados, a passagem também foi efectuada em parques privados de estacionamento e algumas passagens exclusivas a peões. O uso de dois GPS prende-se com o facto de um ser obrigatório, para através do Kismet, registar a latitude e longitude dos locais onde é detectado sinal de redes sem fios; o segundo, é usado para orientação em tempo real do percurso e para medições médias de distâncias e velocidades das passagens em todas zonas.

	Velocidade média (km/h)		Distância percorrida (km)	
Zona	**1ª passagem**	**2ª passagem**	**1ª passagem**	**2ª passagem**
1	15,7	15,3	8,2	7,4
2	14,7	15,2	13,9	13,2
3	13,6	14,6	14,2	13,7
4	14,2	15,1	15,0	13,7
5	15,7	15,1	7,9	7,1
6	13,6	13,0	19,2	17,6
7	13,5	12,8	14,2	12,8
8	14,3	15,2	5,4	5,0
Total	-	-	98,0	90,5
Média	14,3	15,1	14,1	13,0

Tabela 4.3 - Valores médios de velocidades e distâncias percorridas no *warbiking.*

A tabela 4.1, mostra os valores obtidos pelo segundo GPS aquando a localização dos pontos. É visível que a velocidade média corresponde a uma velocidade moderada, com uma média de 14.7Km/h, de forma a obter o menor erro possível na recolha dos dados. Quanto à distância percorrida, esta apresenta em valores totais uma diferença de 8 km devido ao facto de na primeira passagem, apesar de o percurso ser planeado, o desconhecimento de alguns locais bem como o sentido das ruas, fazia com que fosse necessário passar novamente em algum local, ou por vezes voltar atrás, fazendo assim acentuar a diferença em distância entre as duas passagens.

Obtendo a localização dos pontos 802.11 existentes na cidade, a fase seguinte apresenta-se como uma preparação dos dados para o simulador. Essa fase será seguidamente explicada.

4.4.2 Conversão dos dados para o simulador

Esta fase é uma fase intermédia do estudo. Nesta fase é necessário adaptar e transformar os dados recolhidos anteriormente em dados válidos de entrada para o simulador escolhido para o estudo.

Nesta fase todos os pontos foram processados recorrendo ao programa Kismet. Por cada zona e por cada passagem, foi gerado um ficheiro. gpsxml, que correspondendo a um ficheiro xml, é estruturado da seguinte forma:

Exemplo de uma linha de um ficheiro gpsxml:

<gps-point bssid="XX:XX:XX:XX:XX:XX" source=" XX:XX:XX:XX:XX:XX " time-sec="0" time-usec="0" lat="40.637657" lon="-8.658795" spd="0.000000" heading="0" fix="3" alt="5.500000" signal_dbm="-91" noise_dbm="-95"/>

Assim, um ficheiro gpsxml contém essencialmente o MAC do ponto de acesso, a latitude, a longitude e o sinal da captura de todos os pontos de acesso 802.11 capturados. Nestes ficheiros ficam registados todos os valores obtidos, isto porque para o mesmo MAC é possível ter latitude e longitudes diferentes tendo em conta que o sinal varia. Para finalizar esta fase intermédia e para facilitar o processo, reuniram-se todos os ficheiros de todas as zonas, num só ficheiro. Assim, a informação é totalmente processada uma só vez.

Toda a informação foi processada para posterior uso no simulador. Todo o processamento é efectuado através de um *script* desenvolvido em *python*. Devido a várias razões, o simulador usado foi mudado por duas vezes: usou-se o NS-2, o NS-3 e finalmente o *BoostGraph*. Esta alteração teve impacto nesta fase; cada simulador apresenta um *script* próprio (ver próxima secção). Os *scripts* desenvolvidos foram: *gps2ns2.py* [Anexo II], *gps2ns3.py* [Anexo III] e, por último, *gps2BG.py* [Anexo IV]. No entanto, em todos estes, manteve-se sempre inalterada a leitura do ficheiro e a composição da estrutura onde se processava a informação. O *gps2BG.py* foi o *script* usado para o estudo final (este facto prende-se à escolha do simulador, o que será discutido posteriormente).

O processamento pela qual este script se rege é o seguinte: inicialmente todos os pontos são diferenciados pelo seu MAC, onde para cada ponto de acesso, é possível ter diferentes latitudes e longitudes, correspondentes a diferentes sinais de captura. Com diferentes localizações do mesmo ponto, tendo em conta o sinal, é escolhido o melhor sinal

para indicar a localização dos pontos de acesso. Toda esta informação, processada a partir do ficheiro gpsxml, é guardada numa matriz.

Com estes pontos guardados numa matriz, é calculada a distância de todos para todos, para que desta forma se consiga relacionar todos os pontos de acesso tendo em conta a distância entre os mesmos. O cálculo da distância entre dois pontos com os valores de latitude e longitude, é tanto mais complexo quanto mais preciso for. De forma a diminuir os erros, e, assim aumentar a precisão do estudo, foi escolhido a fórmula de *Vicenty* [**55**]. Esta baseia-se no modelo da "terra elíptica", resultando assim no cálculo de uma distância muito próxima da realidade. A aplicação desta fórmula no *script* desenvolvido, apesar da vantagem na precisão do cálculo da distância, implica uma sobrecarga no processamento, devido à sua complexidade.

Neste *script* - gps2BG.py [Anexo IV] – o alcance é o principal argumento de entrada. Este factor é importante, pois a construção do grafo é diferente para cada alcance diferente definido.

Para finalizar, obtidas todas as distâncias entre todos os pontos, são formadas as arestas de um grafo, limitado pelo alcance, isto é, uma aresta só existe do ponto i para o ponto j se a distância for menor ao alcance, e desta forma gera-se uma aresta com um peso igual a distância. Após o cálculo das arestas, este *script* finaliza o processamento gerando quatro ficheiros: o primeiro - *.n - onde é guardado o número de vértices do grafo e o número de arestas do grafo; o segundo - *.edge – onde são guardadas as arestas do grafo; o terceiro - *.weight – onde são guardados os pesos correspondentes às arestas; e finalmente o quarto - *.coor – onde é guardada, para cada nó, a latitude e a longitude.

Obtendo estes dados processados e guardados nos ficheiros correspondentes, esta fase é concluída. Agora com estes dados, é possível proceder à simulação, e posterior caracterização da rede não planeada mapeada.

4.4.3 Extracção de dados para caracterização da rede

A terceira e última fase do método estruturado para este estudo, foi a que se apresentou com maior dificuldade. O simulador escolhido inicialmente apresentava nesta fase problemas significativos. Este problema será abordado e explicado no assunto seguinte.

Em consequência das duas mudanças do simulador, esta última fase foi a que mais sofreu alterações. No entanto, com a escolha final remetendo-se à transformação da rede num grafo, foi possível caracterizar a rede como era pretendido.

O simulador constrói o grafo com base nos ficheiros que o *script gps2BG* gera. Calcula para cada vértice todos os caminhos para todos os vértices, isto é, calcula o caminho de N vértices para N vértices. O algoritmo de Dijkstra é o algoritmo responsável para o cálculo do caminho mais curto entre todos os vértices. Este algoritmo, para além da distância do caminho mais curto, devolve uma lista de antecessores; desta forma, com auxílio de uma função recursiva, é possível obter o caminho mais curto correspondente à distância mais curta entre dois vértices.

Posto isto e através destes dados foi possível caracterizar a rede, embora não usado propriamente um simulador de rede. Deste simulador foi possível obter dados relativamente à conectividade da rede segundo o alcance estipulado na fase anterior. Esta conectividade diz respeito ao número de nós à qual cada nó consegue estabelecer ligação a 1 hop, 5 hops, 10 hops e ao número máximo de hops. Com estes dados estatísticos foi possível apresentar os resultados deste estudo e com uma análise crítica, tirar conclusões sobre o estudo.

4.5 Simulador

O maior problema para todo o estudo foi a escolha do simulador, tal como foi dito anteriormente. Antes da descrição da problemática e de qualquer simulador é necessário indicar as condições em que a simulação terá de correr.

De maneira a garantir o máximo de rentabilidade para o simulador, usou-se uma máquina específica só para o facto. Esta máquina encontrava-se no Instituto de Telecomunicações de Aveiro, onde todo o trabalho era feito remotamente. As especificações técnicas da máquina foram as seguintes:

- **S.O.** Linux Ubuntu 9.04
- Intel Core 2 Duo 6300
- 4096 Mb RAM DDR2

Seguidamente serão apresentados todos os simuladores testados e utilizados, e, num estudo comparativo, será explicada a escolha final do simulador. Os simuladores inicialmente escolhidos foram o *Network Simulator* 2 (NS-2) e o *Network Simulator* 3 (NS-3). No entanto, nenhum destes foi utilizado na simulação final.

4.5.1 NS-2

Inicialmente todo o trabalho foi desenvolvido para usar o Network Simulator 2 [**56**], porque é um simulador bastante completo, grátis, e inicialmente o número de pontos de acesso 802.11 esperado não era muito elevado (número total de nós encontrados: 5478).

Após correr as primeiras simulações teste neste simulador, deparou-se com uma situação que não se conseguia resolver: um elevado uso de memória da máquina em simulações testes com 100 nós, com tempo total de simulação de dias. Este uso excessivo de memória talvez se devesse ao facto de este simulador usar oTcl como linguagem de programação, que sendo uma linguagem interpretada se apresenta em desuso e com problemas de desempenho consideráveis.

Para avaliação dos simuladores, o NS-2 foi programado para que nas simulações, usando o protocolo *Ad-hoc On-demand Distance Vector* (AODV), fossem enviados pequenos pacotes *Transmission Control Protocol* (TCP), um a um, de todos os nós para todos os nós, de forma a verificar conectividade entre todos os nós da rede. Inicialmente, foram concedidos 120 segundos para estabilização do protocolo. Assim, o tempo de simulação calculado foi, em segundos: 120 (tempo de espera inicial) + **(**(10 (tempo de inicio da aplicação) + 5 (tempo de envio) + 5 (tempo de intervalo)) * 5478 *5478 (envio de pacotes de todos os nós para todos os nós)**)** + 120 (tempo de espera final), perfazendo um total de 600169920 segundos (≈6946 dias). Esta simulação, em tempo real, demoraria cerca de 113 dias. Facilmente, foi possível concluir que este tempo seria impossível aceitar. E para além deste problema, o uso excessivo de RAM era outra complicação. O simulador apresentava-se estável durante alguns minutos com cerca de 15% de RAM dispendida e logo depois a RAM subia em média 0.2 % por minuto (ver tabela 4.4). De forma a corrigir os dois problemas, houve necessidade de mudar o simulador.

4.5.2 NS-3

Perante a situação anteriormente apresentada, foi necessário analisar os vários simuladores existentes no mercado para simulações em grande escala. Tendo em conta o estudo [**57**], optou-se pelo NS-3 [**58**] , pois neste fez pouco uso de memória, factor predominante, e apresentava-se muito estável e com desempenho muito bom em relação ao NS-2. Para além de se apresentar muito completo, como o NS-2, o NS-3 está actualizado em vários módulos e usa C++ como linguagem de programação, é mais uma vantagem em relação ao NS-2.

Posta esta alteração, desenvolveu-se o código para o novo simulador escolhido, NS-3 [Anexo VII]. O objectivo era o mesmo que o anterior: correr inicialmente simulações testes apenas com 100 nós e com o envio de pequenos pacotes, mas neste caso UDP e com o protocolo OLSR. Foi necessário modificar o protocolo de *routing* visto que o NS-3 não suportava o protocolo AODV. Para que o protocolo estabilizasse, tal como no NS-2, os primeiros 120 segundos no início da simulação eram descontados, começando de seguida o envio de pacotes. O tempo de simulação manteve-se inalterado em relação ao tempo de simulação no NS-2.

O NS-3, ao contrário do NS-2 apresentou-se muito concistente em relação à memória dispendida, pois subiu até cerca dos 4% e depois manteve-se estável (ver tabela 4.4). Mas se com a mudança para o NS-3 se conseguiu resolver os problemas de memória, não se conseguiu resolver um problema importante, o tempo que demorava a simulação em tempo real, pois estimando o valor da simulação em tempo real, apenas se obteria valores passados 228 dias.

Com as estimativas apresentadas é agora possível comparar os dois simuladores e demonstrar que ambos não são soluções para o pretendido.

4.5.3 Análise conclusiva dos simuladores

O funcionamento dos dois simuladores foi exposto anteriormente. Com os dados obtidos é possível concluir sobre a utilização ou não dos mesmos, é apresentado numa tabela resumo com os valores anteriormente referenciados.

Simuladores	Tempo real (min)	Tempo simulado (min)	Tempo real necessário (dias)	Memória máxima dispendida (%)
NS - 2	1589 (*)	27	113	42,7%
NS - 3	3818 (*)	32	228	4,4%

(*) Tempo à qual a simulação foi interrompida

Tabela 4.4 - Valores comparativos dos vários simuladores.

Na tabela 4.4, são apresentados os valores referentes aos tempos e ao uso da memória das diferentes simulações. Como é possível concluir, a mudança de simulador apresentou-se como uma enorme vantagem em relação à memória, mas continua impraticável em relação aos tempos de simulação. Com estes valores foi necessário repensar a parte de simulação deste trabalho. Não obstante dificuldades encontradas anteriormente, preferiu-se sempre um simulador real a alguma adaptação ao estudo. Uma hipótese para resolver este problema, era usar programação distribuída, e dividir todo o processamento por várias máquinas. No entanto, devido ao facto de ser necessário dispender várias máquinas para conseguir obter resultados, procuraram-se novas soluções, deixando esta opção para última. As soluções passavam por conseguir obter resultados em poucos minutos, de forma a conseguir perceber se existiam erros na programação do simulador, ou inclusive nas fases anteriores do estudo.

Tal não se tinha conseguido com os simuladores apresentados, então, surgiu uma solução: transformar a rede num grafo. O processamento necessário, para se conseguir caracterizar a rede a partir desta adaptação, era muito menor.

Nota: todo o código desenvolvido para o NS-2 e NS-3, encontra-se disponível nos anexos II e III.

4.5.4 Descrição do simulador final

Expondo anteriormente toda a fundamentação da escolha do simulador, nesta secção apenas será caracterizado o código desenvolvido para o simulador final. Tal como foi

explicado anteriormente, para conseguir simular segundo a topologia encontrada, transformou-se a rede num grafo correspondente.

Posto isto, para desenvolver o código necessário foi preciso recorrer ao Boost Graph [**59**] , uma livraria grátis para a linguagem de programação C++, que permite construir e operar sobre grafos de várias formas, apresentando códigos de fácil uso e com um desempenho bastante considerável. A rapidez do simulador desenvolvido, perante o processamento e em comparação aos simuladores utilizados anteriormente, foi factor decisivo.

Alcance (m)	Duração da simulação	Registo máximo de memória (%)
50	1m24s	16.7
100	2m55s	23.5
150	4m57s	24.7

Tabela 4.5 - Valores de simulação para vários alcances dos APs.

A tabela 4.5 apresenta valores obtidos com o simulador programado, com diferentes alcances. É visível que o alcance é directamente proporcional ao tempo, isto é, quanto maior o alcance maior é a duração da simulação. Este facto deve-se ao aumento do número de arestas que é necessário processar para cada alcance. Sendo estas construídas segundo o alcance definido inicialmente, quando se aumenta o alcance, o número de arestas também aumenta pois com um alcance maior é mais provável encontrar mais nós vizinhos. Este procedimento para criar as arestas é tratado na fase intermédia do método aplicado, pelo *script gps2BG*. O simulador apenas constrói um grafo, cujas arestas já se encontram anteriormente definidas. Este simulador foi completamente programado em C++ e graças à livraria disponível para esta linguagem, Boost Graph, esta construção é um processo fácil.

Esta livraria já incluía vários algoritmos optimizados para operar sobre um grafo definido, nomeadamente, algortimos para obter o caminho curto de um vértice para outro vértice do grafo. Desta forma, tendo o grafo, o algoritmo de Dijkstra foi usado para conseguir caracterizar a topologia da rede encontrada.

Em suma, o simulador final apresentou-se viável e consistente. Para que tal acontece-se, houve vários cuidados com a memória e com o código implementado, de forma a não ter nenhum problema como o verificado nos simuladores anteriores.

4.5.5 Descrição do cenário do simulador

O simulador programado, não contempla definições de rede, mas sim definições do grafo no qual a topologia da rede foi transformada. Então, foi definido inicialmente o tipo de grafo: grafo que possui arestas sem direcção e que têm um peso que é a distância definida entre os dois nós. Esta definição está descrita no excerto de código seguinte.

Código do simulator.cpp:

```
...
typedef adjacency_list < listS, vecS, undirectedS,no_property, property < edge_weight_t, float > > graph_t;
…
graph_t g(num_nodes);
…
```

Criando o grafo, torna-se agora possível com esta livraria operar sobre o mesmo. Após isto foi necessário adicionar arestas e pesos às arestas do grafo, através da leitura dos ficheiros de saída do *script gps2BG*, concluindo assim a construção do grafo.

Outro aspecto a realçar neste simulador, é o facto de usar o algoritmo de Dijkstra para o cálculo de todos os caminhos mais curtos entre todos os vértices do grafo. Dispondo de vários algoritmos para este tipo de cálculo, o Dijkstra, devolve uma matriz com distâncias e uma matriz de antecessores, e esta última, permite o cálculo de todo o trajecto para chegar de um vértice de origem a um vértice de destino, com ajuda de uma função recursiva.

Posto isto, torna-se necessário demonstrar como este algoritmo de Dijkstra funciona no nosso simulador, para que seja compreendido o cenário final com que será possível caracterizar a topologia da rede. O algoritmo apresenta argumentos de entrada para poder calcular o caminho curto. São eles:

- *g:* o grafo definido anteriormente;
- *s:* vector que contêm todos os vértices;
- *weightmap:* vector que contem os pesos de todas as arestas;
- *indexmap:* vector que contêm o índice de todos os vértices destinos a partir de um vértice de origem;
- *less<int>():* é a função utilizada para comparação entre distâncias;

- *closed_plus<int>():* é a função utilizada para acumular as distâncias de um caminho curto;
- *(numeric_limits<int>::max)():* é o número maior possível para um inteiro, isto é, quando não existe ligação entre dois vértices é atribuída uma distância infinita;
- *0:* é o número mais pequeno possível para a distância.

Com estes argumentos é possível o cálculo de todos os caminhos curtos. Este cálculo é demonstrado seguidamente no excerto do código do simulador, onde **k** é o índice do vértice em cálculo.

Código do simulator.cpp:

```
...
dijkstra_shortest_paths(g, s[k], &p[k][0], &d[k][0], weightmap, indexmap,less<int>(),
                closed_plus<int>(),(numeric_limits<int>::max)(), 0,default_dijkstra_visitor());
…
```

Desta forma, as duas matrizes de saída são:

- *d:* matriz dos todos os caminhos curtos para todos os vértices;
- *p:* matriz de todos os antecessores no caminho curto de todos os vértices.

Finalmente, obtêm-se com o cenário final: um grafo com 5478 vértices e diferentes arestas, conforme o alcance definido anteriormente, e com a informação de todos os trajectos de um vértice de origem para um vértice destino. Este grafo com estas informações será agora estudado e caracterizado para que, por forma análoga se consiga caracterizar a possível topologia dos pontos de acesso da cidade de Aveiro.

4.6 Avaliação

Com dados obtidos no cenário anterior, é agora possível agora obter alguns valores para caracterizar a rede que se apresenta com a topologia simulada no grafo. Ainda através destes é possível estimar outros valores, nomeadamente a largura de banda disponível para cada nó atendendo ao facto da distância e a probabilidade de erro na transmissão dos pacotes.

O simulador devolve valores de conectividade globais para um alcance estipulado, isto é, em média, com um dado alcance, qual é a conectividade dos nós com múltiplos hops, com 1 hop,

com 5 hops e com 10 hops. Estes valores de conectividade são importantes para perceber como se encontra esta topologia, isto é, onde existe lacunas e sendo assim, em que aspectos será necessário reforçá-la de forma a garantir conectividade em qualquer ponto da cidade. Outros aspectos serão ainda apresentados, tais como o espectro da cidade, estatísticas sobre segurança e ainda aspectos relacionados com a densidade populacional da cidade. Todos os resultados serão seguidamente descritos e analisados, para posterior conclusão do estudo.

4.7 Resultados

Os resultados serão nesta fase apresentados segundo uma ordem lógica. Estes serão divididos em várias categorias, onde em cada uma é descrito como se obteve o resultado, seguido de uma breve análise.

4.7.1 Número de nós

O primeiro valor a destacar nesta secção é o número total de pontos de acesso localizados na área de estudo da cidade de Aveiro: **5478** (ver figura 4.5). É um valor considerável. No entanto, tendo em conta, a que a área da cidade estudada, refere-se ao centro da cidade, onde existe mais estudantes, mais escritórios, mais escolas e mais casas de habitação, este valor é compreensível. Este valor foi obtido através do processamento gerado pelo *script gps2BG* [Anexo IV].

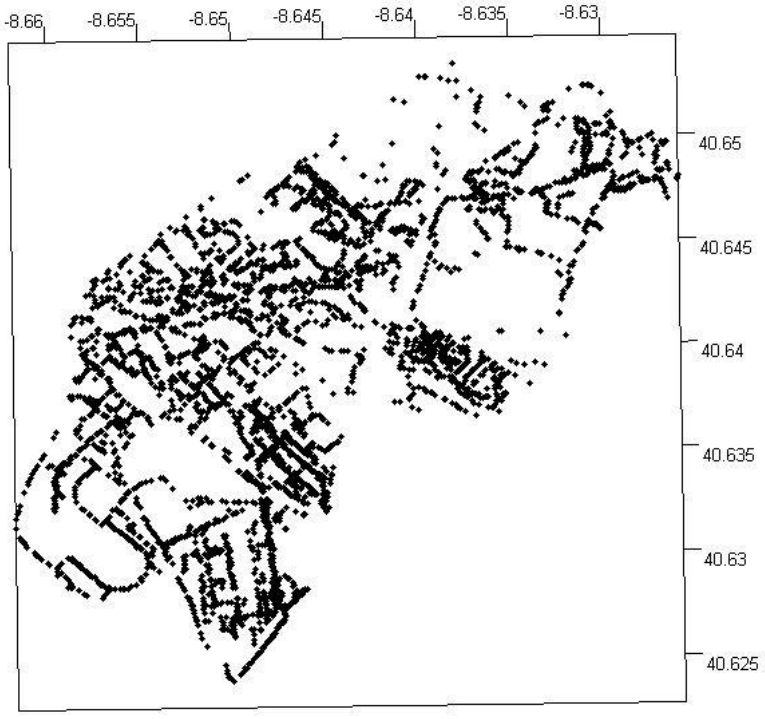

Figura 4.5 - Mapeamento dos pontos de acesso localizados em ambiente de simulação.

Na figura 4.5 está representado o mapeamento da cidade de todos os pontos de acesso localizados. É perceptível que existe uma maior densidade de pontos de acesso em algumas zonas em relação a outras. Isto deve-se ao facto de alguma zonas serem mais

movimentadas em vários sectores do que outras zonas, que são apenas zonas de habitação e/ou com pouco comércio. Para se conseguir perceber este facto, seguidamente é apresentado detalhadamente o número de pontos localizados por zona.

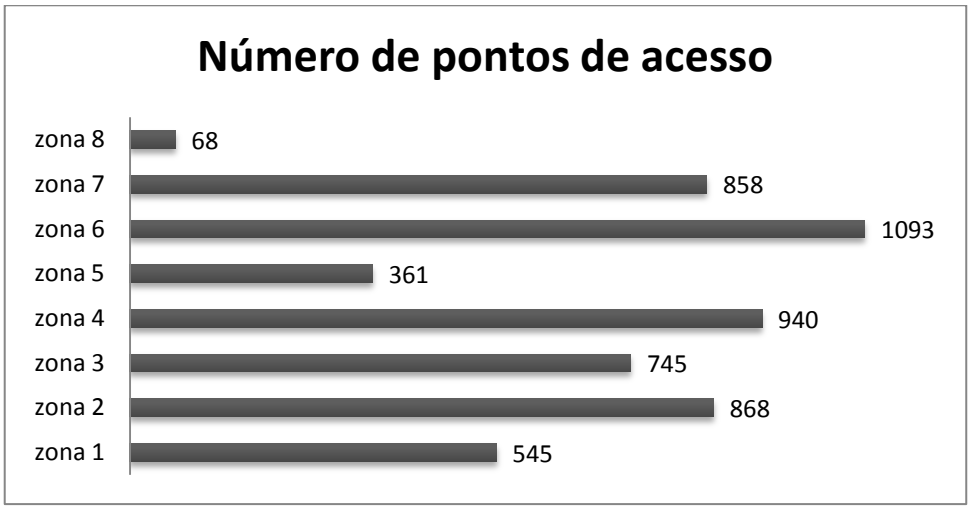

Figura 4.6 - Número de pontos de acesso por zona.

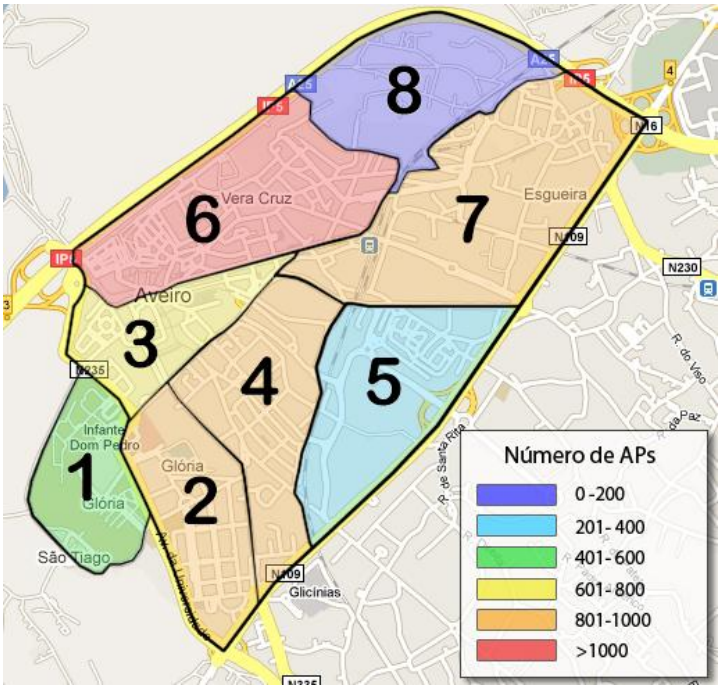

Figura 4.7 - Mapa com a densidade de APs por zona

As figuras 4.5 e 4.6, mostram o número de pontos de acesso, por zona, perfazendo um total de 5478 nós. As zonas 2,4 e 6 são, como se pode verificar, as zonas com um

maior número de pontos de acesso. Estes valores são aceitáveis, visto que estas três zonas referem-se ao centro de duas das principais freguesias deste estudo: Glória e Vera Cruz.

4.7.2 Alcance vs. Número de arestas

Tendo em conta que a topologia da rede foi representada num grafo, outro aspecto importante é qual a relação existente entre o alcance escolhido para a simulação, e o número de arestas geradas para a definição do grafo. Esta relação é importante para compreender que existe uma relação directa entre os dois aspectos da topologia.

Alcance (m)	Número de arestas total	Número médio de arestas por vértice
40	36328	6
50	50467	9
60	67143	12
70	86314	15
80	104469	19
90	131701	24
100	158267	28
110	186053	33
120	216014	39
130	247966	45
140	281719	51
150	317439	57

Tabela 4.6 - Alcance vs. Número de arestas.

A tabela 4.6 representa a relação existente entre o alcance e o número de arestas. É perceptível que quanto maior é o alcance, maior é o número de arestas total e, por consequência, o número médio de arestas por vértice, ou nó neste caso, também é maior. Para uma vizualização mais directa destes valores seguidamente é apresentada a linha de tendência dos valores da figura 4.8.

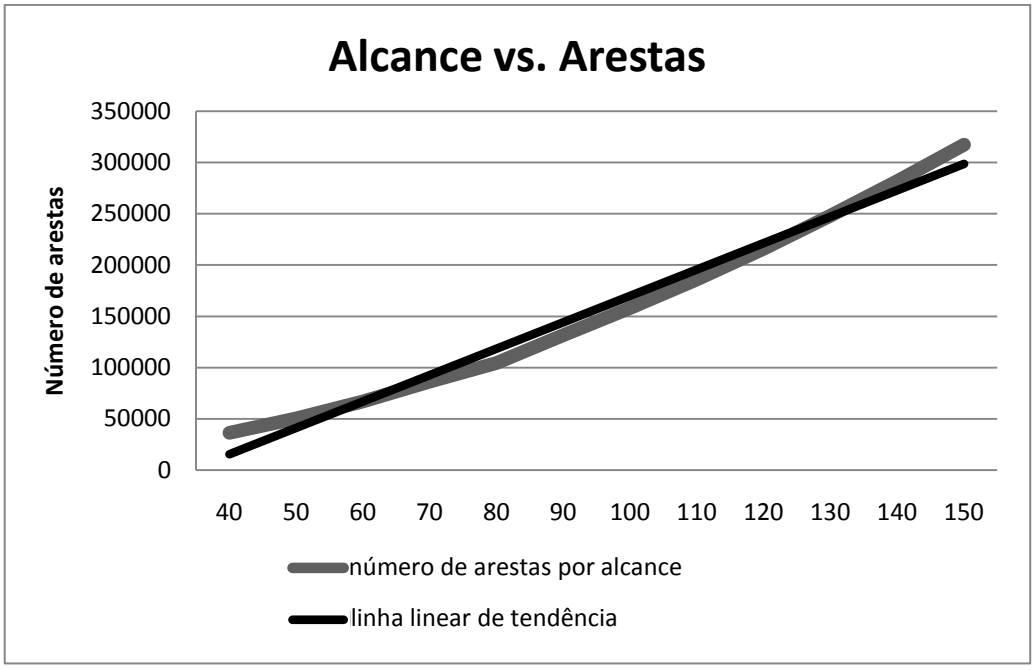

Figura 4.8 - Linha de tendência onde representa a relação entre o alcance e o número de arestas do grafo correspondente à topologia da rede encontrada.

O gráfico representado na figura 4.8, mostra a linha de tendência em preto do número de arestas total correspondente a cada alcance. A proporcionalidade directa encontra-se expressa nesta como se queria mostrar.

Neste estudo para posteriores análises, fixou-se o alcance em 50 metros, pois é considerado um alcance realista, tendo em conta os vários tipos de adversidades, nomeadamente a estrutura das casas e dos edifícios, tanto de habitação como para serviços públicos. Para além deste factor, a radiação e o limite de potência estabelecido, são outros dois pontos para estabelecer o alcance de **50** metros como limite. Para os 50 metros de alcance, as arestas apresentavam um peso médio (neste caso distância) de **29,05** metros.

Estes dados foram obtidos através da informação processada no *script gps2BG* [Anexo IV].

4.7.3 Distância média entre os nós

Se anteriormente foi apresentada a variação existente entre o alcance e o número de arestas, nesta fase é apenas apresentado, segundo a topologia encontrada, a distância

média entre cada vértice do grafo, na prática a distância média entre os nós da rede. Sendo assim, em média que cada nó dista de um outro é de **1231** metros. Este valor ajuda a perceber que apesar de existirem zonas onde existe uma densidade considerável, existem outras que contra balançam com distâncias bem maiores fazendo assim com que a distância média seja relativamente grande. Esta distância média, acabar por idealizar a ideia errada, de que os nós se encontram muito afastados, no entanto, este valor é apenas uma média obtida da soma de todas as distâncias, de todos os pontos para todos os pontos. Ao analisar detalhadamente os dados, verifica-se que este tem em conta, tanto distâncias de nós de uma zona para outra zona oposta na cidade, bem como valores da mesma zona com distância na ordem das unidades.

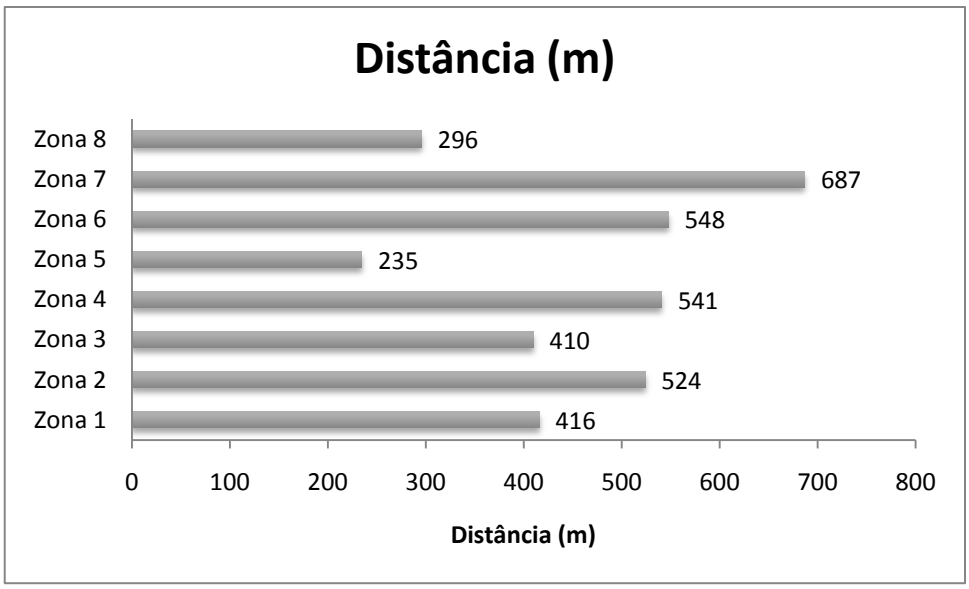

Figura 4.9 - Distância média entre nós por zona

O gráfico da figura 4.9 mostra-nos a distância média entre nós por cada zona. Este gráfico apresenta-nos valores mais concretos, pois refere-se a cada zona, verificando-se como seria de espera uma distância média menor em relação à distância média total de todas as zonas.

Os valores apresentados na figura 4.9 foram calculados através do *script distances* [Anexo X], programado para este facto.

4.7.4 Indicadores estatísticos genéricos vs. Número de pontos de acesso

Um aspecto analisado neste estudo, e deveras importante é relacionar a área de estudo e sua densidade populacional com o número de pontos de acesso localizados na mesma área. Assim sendo, e tendo em conta os valores apresentados na tabela 4.7 e o número de nós localizados, foi possível estimar dois novos valores importante que ajudam a caracterizar a topologia da rede encontrada.

				Novos valores estimados	
	Área de estudo (Km^2)	População relativa à área de estudo (Hab)	Número de pontos de acesso (PA)	Relação PA por habitante (Hab/PA)	Relação PA por Km^2 (PA/Km^2)
Total	12	22 250	5478	**4**	**457**
Zona	(*)	(*)			
1	1	1 500	545	**3**	**545**
2	1,5	3 500	868	**4**	**579**
3	1	3 500	745	**5**	**745**
4	1,5	4 500	940	**5**	**627**
5	1,5	1 500	361	**4**	**241**
6	2	5 000	1093	**5**	**547**
7	2	2 500	858	**3**	**429**
8	1.5	250	68	**4**	**45**
					(*) Valores estimados

Tabela 4.7 - Valores estatísticos sobre a topologia da rede.

A tabela apresenta os novos valores estimados sobre a topologia da rede. Estes valores ajudam a compreender como a topologia da rede está disposta actualmente. A densidade de pontos de acesso está bem presente. O facto de existirem aproximadamente 457 pontos de acesso por Km^2, representa uma grande densidade de terminais num espaço considerável. Para conseguirmos perceber melhor este apecto, a relação habitante por ponto de acesso explica melhor a topologia encontrada: tendo em conta que cada habitação tem em média 4 pessoas, então podemos estimar que existe um PA por habitação. Também visível na tabela que zonas com mais habitantes (2,4 e 6) apresenta um número de habitantes por PA e uma área maior.

4.7.5 Alcance vs. Conectividade

Um dos resultados, que essencialmente se pretendia obter neste estudo era saber qual a taxa de conectividade média para uma rede com esta topologia. E tal como foi descrito anteriormente, isto foi possível através da analogia da topologia da rede com um grafo. Ao simular os dados com um dado alcance, os valores que são extraídos directamente são: a conectividade média a um *hop*, a cinco *hops*, a dez *hops* e a múltiplos *hops*. Estes valores serão o maior complemento para a caracterização desta rede, pois indicarão indirectamente até que ponto a topologia da rede se apresenta prestável para os diferentes serviços que se pretendem implementar.

Recorrendo a um *script* desenvolvido com intuito de simular os dados recolhidos, e através da consulta dos ficheiros *.results correspondentes a cada alcance obteve-se os resultados apresentados seguidamente.

	Conectividade média (número de *hops*)			
Alcance (m)	**1 *Hop***	**5 *Hops***	**10 *Hops***	**Multiplos *Hops***
40	10	66	161	2341
50	14	103	268	3217
60	17	149	385	4399
70	22	190	491	5288
80	26	232	606	5294
90	31	277	726	5371
100	36	324	850	5411
110	41	364	949	5411
120	46	409	1074	5415
130	51	454	1189	5463
140	56	492	1299	5469
150	61	527	1380	5473

Tabela 4.8 - Alcance versus conectividade.

Na tabela 4.8, estão representados os valores obtidos nas simulações. Verifica-se que quanto maior for o alcance maior é a conectividade em cada ponto de acesso. Esta relação directa faz sentido pois, ao aumentar o alcance, consegue-se obter mais pontos que

tenham uma distância menor ou igual ao alcance, e desta forma, consegue estabelecer uma nova aresta no grafo.

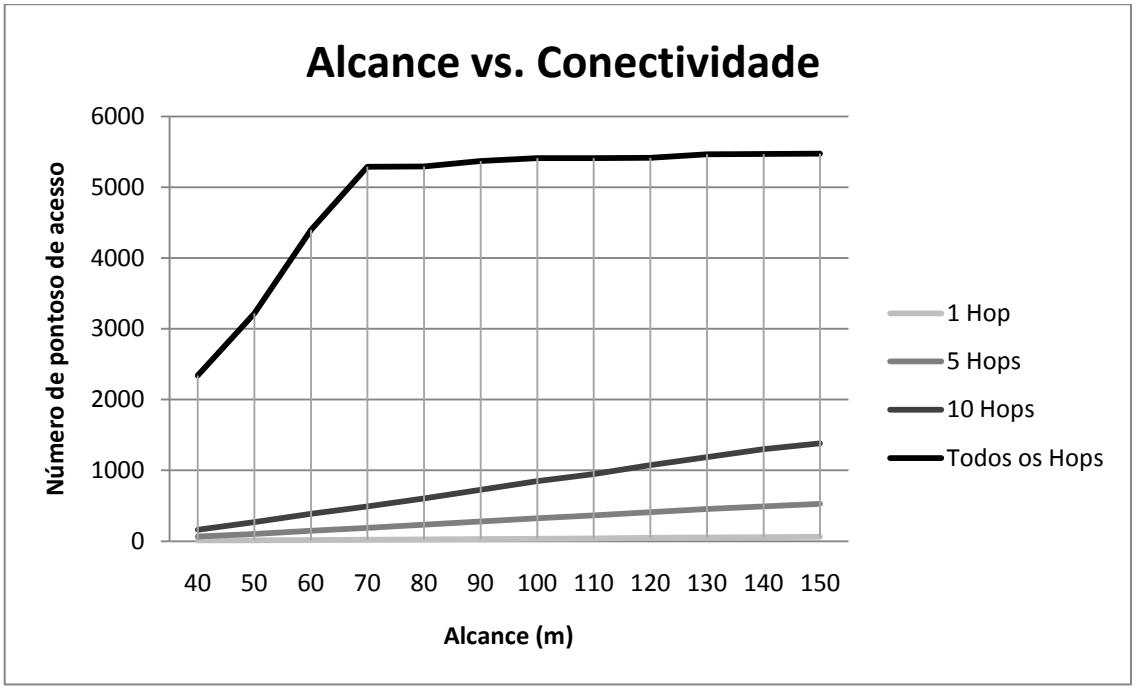

Figura 4.10 – Valores de alcance em relação à conectividade média em qualquer ponto da rede.

A figura 4.10 confirma a relação directa entre o alcance e a conectividade média descrita anteriormente. No entanto é visível pelo gráfico representado na figura 4.10, que a partir de um alcance crítico existe um limite que é estacionário, isto é, a partir dos 90 metros de alcance, verifica-se que um nó da rede em qualquer ponto da topologia apresenta uma possibilidade de ligação a outro ponto de acesso (conectividade média) de cerca de 98,0%, que é pouco maior que os 99,9 % de um alcance de 150 metros.

Para o alcance estabelecido como padrão para este estudo, 50 metros, obteve-se uma conectivade média total de aproximadamento 58%. Isto significa que em média um nó da rede só consegue estabelecer ligação com 58% de outros nós na rede. Assim existem pontos à qual não é possível estabelecer ligação. Este pontos são caracterizados como ilhas, e, para estes é necessário garantir conectividade. Para cada ilha é necessário um *gateway* específico para garantir conectividade para com os outros pontosde acesso.

Sucintamente, pode-se verificar, que com um alcance razoável, 50m, os resultados obtidos encontram-se dentro dos parâmetros aceitáveis. Com um posicionamento de alguns

pontos de acesso estratégicos na topologia seriam possíveis provavelmente melhores resultados. No capítulo final, estes valores serão discutidos novamente para que se possa concluir sobre os resultados aqui apresentado.

4.7.6 Largura de banda vs. Alcance

Um dos resultados com maior impacto neste tipo de estudo é a largura de banda disponível para um cliente que pretenda estabelecer ligação a um dado nó para obter ligação a internet. Com estes dados é possível estimar que serviços esta rede poderá operar, e consequentemente, que serviços poderá oferecer aos seus potenciais clientes.

Em primeiro lugar e tendo em conta que não foi possível realizar uma simulação de rede, os valores obtidos para esta secção são estimados através de fórmulas disponíveis em [**60**]. Em segundo lugar para além das fórmulas utilizadas, foi necessário estabelecer algumas restrições. As restrições consideradas foram:

- *TX Power*: 15dBm;
- *TX/RX Antenna Gain*: 3.5dBi;
- *Chipset* no receptor: Atheros 5006 (Ubiquiti Super Range Express), [**50**];
- *Scattering Exponent:* 4;
- *Allowed Loss*: 0 dB;
- *IEEE 802.11g OFDM.*

Tendo em conta, estas restrições, os valores de largura de banda para cada alcance são apresentados na tabela 4.9

Alcance (m)	Largura de banda (Mbps)
40	36
50	24
60	18
70	12
80	6
90	2
100	1

Tabela 4 9 - Largura de banda vs. Alcance

Na tabela 4.9 estão apresentados os valores de largura de banda relativos a cada alcance. É visível que estes apresentam uma relação inversa, isto é, quando o alcance aumenta, a largura de banda diminiu. Para os alcances superiores a 100 metros, não foi possível determinar a largura de banda, visto apresentar-se num valor muito pequeno, sendo impossível de usar efectivamente.

É importante perceber estes valores. Com esta estimativa, entende-se que o receptor esta no limite do alcance, isto é, no limite do raio onde a emissão de sinal é propagada. Posto isto, com um alcance de 40 metros, o receptor encontra-se mais próximo do emissor, sofrendo menor interferência, e obtendo assim uma largura de banda consideravelmente boa. No entanto, com um alcance de 100 metros, (que tem uma conectividade média maior) a nível da largura de banda disponível para o cliente, apresenta-se como uma desvantagem, pois permite ao receptor afastar-se do emissor, mas diminuindo em fortemente a sua largura de banda.

4.7.7 Ilhas

Um aspecto importante para caracterização desta topologia, é saber ao certo quantas ilhas existem para um determinado alcance. As ilhas são conjuntos de nós da rede que não têm acesso a nenhum outro nó da rede. Isto acontece devido ao facto de se encontrarem isolados dos outros nós da rede. Tendo em vista a projecção de uma rede sem fios em malha, é necessário garantir que todos os nós tenham acesso à internet, sendo portanto, necessário no mínimo, que consiga obter ligação a um nó que tenha acesso a internet. Como se trata de uma topologia já definida, é necessário optar por outros recursos:

os *gateways*. Estes dispositivos, permitirão que nós que se retratam como ilhas permaneçam em contacto com a rede e essencialmente com a internet.

Tendo em conta um alcance de 50 metros como alcance padrão, na tabela 4.10 estão indicadas as ilhas encontradas.

Número de elementos da ilha	Número total de elementos	Número de ilhas iguais
4129	4129	1
634	634	1
420	420	1
32	32	1
25	50	2
21	21	1
16	16	1
13	13	1
12	12	1
10	20	2
8	8	1
7	7	1
6	18	3
5	30	6
4	20	5
3	18	6
2	8	4
1	22	22
Número total de ilhas		**60**

Tabela 4.10 - Número de ilhas com um alcance de 50 metros

A tabela 4.10 apresenta-nos as ilhas existentes na topologia não planeada encontrada na cidade de Aveiro com um alcance de 50 metros. Esta tabela mostra-nos que existem 60 ilhas, que todos unidos perfazem um só conjunto de 5478 nós. Para esta reunião de conjuntos são necessários 60 *gateways* (um por ilha), permitindo conectividade entre em todos os pontos.

Outro aspecto a ter em conta no caso das ilhas são os pontos isolados. Estes são ilhas específicas, apenas têm um elemento. Para os vários alcances expostos anteriormente, o número de pontos isolados está indicado na tabela 4.11.

Alcance (m)	Pontos Isolados
40	45
50	22
60	16
70	11
80	8
90	5
100	4
110	4
120	2
130	0
140	0
150	0

Tabela 4.11 - Pontos isolados em função do alcance

Como seria de esperar, quanto maior o alcance, menor o número de ilhas, visto que maior é a conectividade em cada nó. É visível pela tabela 4.11 que a partir dos 130 metros de alcances, não existem pontos isolados, diminuindo assim o número de *gateways* necessários. No entanto, aos 50 metros, o alcance padrão estabelecido para o estudo, existem 22 pontos isolados. Este número é um número considerável. Tendo em conta o número de nós, esperava-se um número menor de pontos isolados. No entanto, a grande distância entre os nós da rede especialmente na zona 7 (ver figura 4.7), é um factor crítico para o número de pontos isolados.

Para o cálculo do número de ilhas foi necessário recorrer ao ficheiro *.plot das simulações respectivas a cada alcance. Neste ficheiro estão contidas as informações de conectividade de todos os pontos.

4.7.8 Espectro

Sendo a sobreposição de canais um dos problemas das redes sem fios, também será realçado os resultados obtidos deste estudo nesse campo. Com as captura obtidas e com o *script count_channels* [Anexo IX], foi possível obter a quantidade de pontos de acesso que operavam nos 13 canais disponíveis em Portugal. Nos resultados que serão em seguida apresentados, admite-se a existência de um canal 0, que se refere ao facto de serem *probe networks*, redes a qual ainda não foi possível detectar em que canais operam.

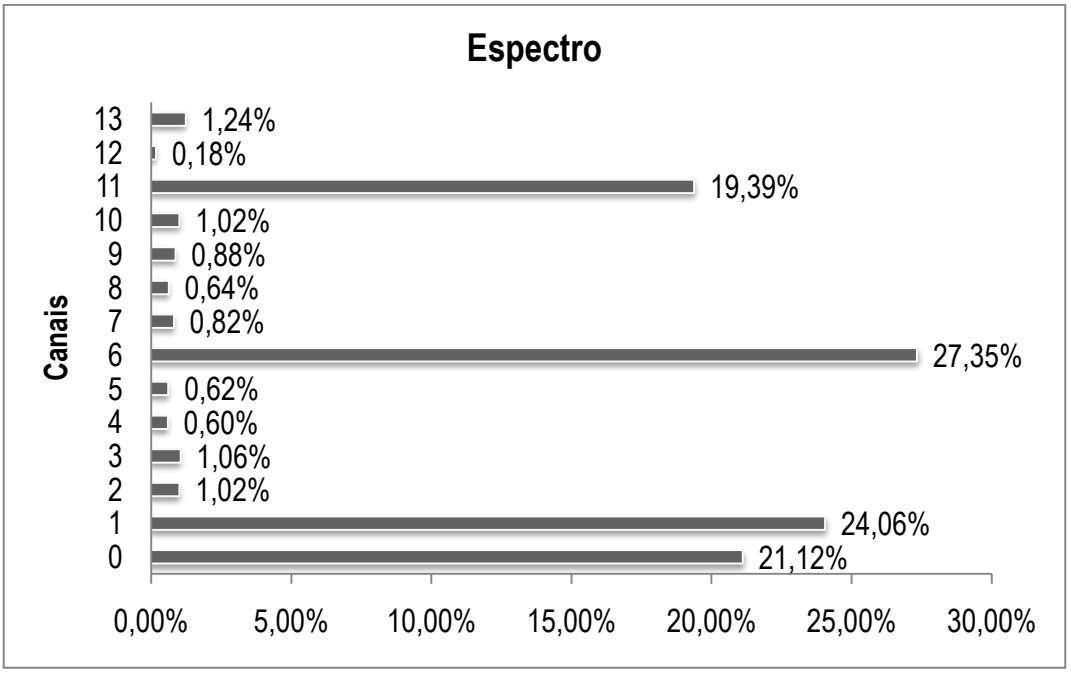

Figura 4.11 - Gráfico representativo do espectro.

A figura 4.11 representa o espectro dos canais de todos os pontos de acesso localizados na cidade de Aveiro. Com este gráfico podemos verificar duas grandes tendências: em primeiro lugar, é perceptível que existe um grande número de PA à qual não é possível detectar em que canais operam, cerca de 21 %; e em segundo lugar existe 3 canais que representam 70% da ocupação do espectro, os canais 1,6 e 11. Este facto é importante, pois tal como foi provado nos capítulos 2 e 3, este são canais que não se sobrepõe, o que pode significar um bom desempenho em largura de banda disponível para os utilizadores desses nós. No entanto, se existirem zonas, onde a densidade de PA, coincidir com a utilização dos mesmos canais, pode levar a um efeito contrário do desejado, uma diminuição abrupta da largura de banda, tal como foi comprovado no capítulo três.

4.7.9 Segurança

Nos dias de hoje, muita informação confidencial circula pela Internet. Cada vez mais é necessário ter em atenção questões de segurança relacionadas com as redes sem fios. Neste estudo decidiu-se também apresentar características de segurança dos pontos de acesso encontrados.

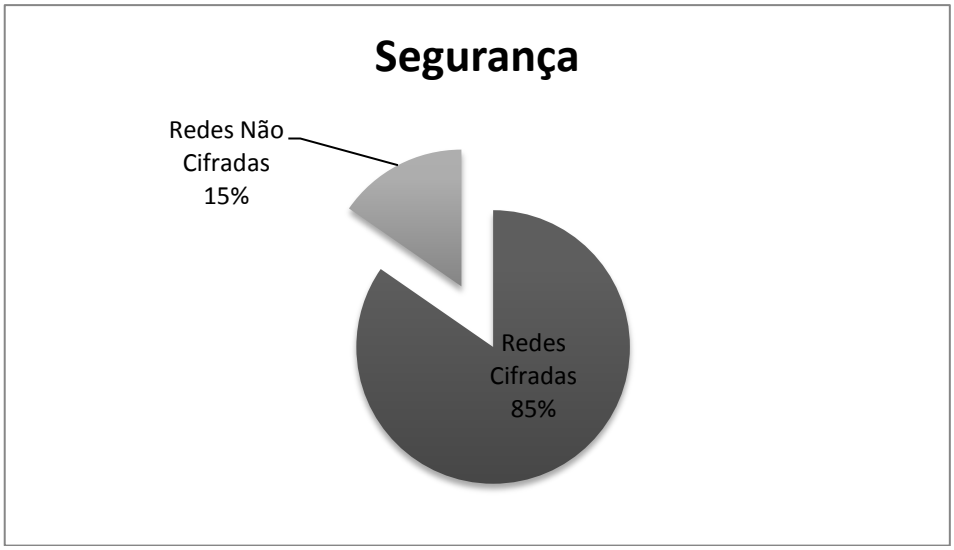

Figura 4.12 - Valores de encriptação relativos aos pontos de acesso localizados na cidade de Aveiro.

A figura 4.12, mostra-nos os valores de redes cifradas e não cifradas obtidos aquando a localização de todos os pontos de acesso da área em estudo. De 5478 pontos localizados, 4637, (cerca de 85%) apresentam segurança activa, cifrando os seus dados. No entanto, os restantes 841 (aproximadamente 15 %) apresentam lacunas neste sentido. A percentagem recolhida de redes cifradas, apresenta-se com um bom número, mas, o ideal seria que, por questões de segurança, todos os pontos de acesso, apresentassem normas de segurança implementadas. Esta estatística foi obtida através do *script security* [Anexo VIII].

4.7.10 Velocidade de Transmissão

Os serviços que futuramente poderão prestados aos habitantes da cidade de Aveiro dependem da velocidade de transmissão. Assim, as velocidades de transmissão encontradas na topologia não planeada dos pontos de acesso da cidade de Aveiro são também importantes para a caracterização da mesma.

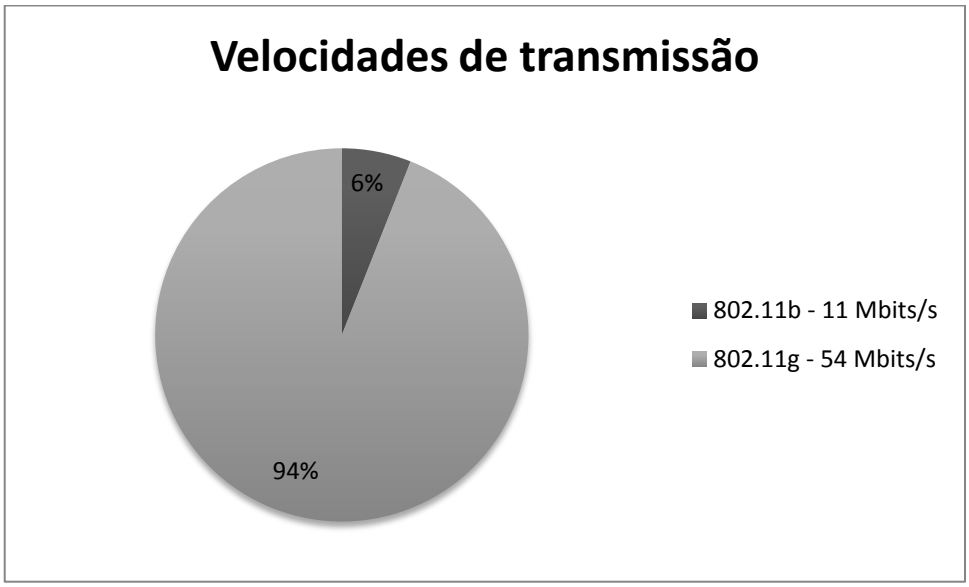

Figura 4.13 - Velocidades de transmissão detectadas nos pontos de acesso localizados na cida de Aveiro

A figura 4.13 apresenta-nos os valores das velocidades de transmissão que os pontos de acesso localizados na cidade de Aveiro operavam. Num total de 5478 pontos localizados, 5149 funcionam a uma velocidade máxima de 54 Mbits/s (802.11g), os restantes 329 funcionam a uma velocidade máxima de 11Mbits/s (802.11b).

4.7.11 Modos de funcionamento

Como foi explicado no capítulo dois existem dois modos de funcionamento das redes sem fios: *ad-hoc* e infra-estruturado. Os modos de funcionamento também fazem parte da caracterização da topologia não planeada encontrada na cidade de Aveiro.

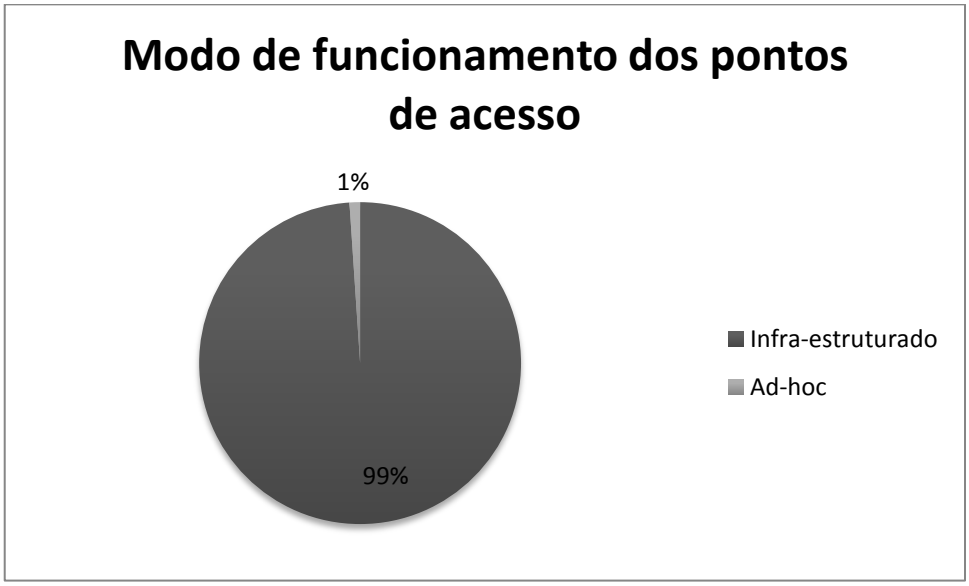

Figura 4.14 - Modos de funcionamento dos pontos de acesso localizados na cidade de Aveiro

A figura 4.14 mostra-nos as percentagens dos pontos de acesso localizados na cidade de Aveiro que funcionam no modo *ad-hoc* e no modo infra-estruturado. De 5478 pontos localizados, apenas 58 funcionam no modo *ad-hoc* representando 1% do número total de pontos localizados. Todos os restantes pontos localizados (5420), funcionam no modo Infra-estruturado (99%).

4.7.12 SSID

Por último, os SSID encontrados nos pontos de acesso localizados na cidade de Aveiro foram também caracterizados.

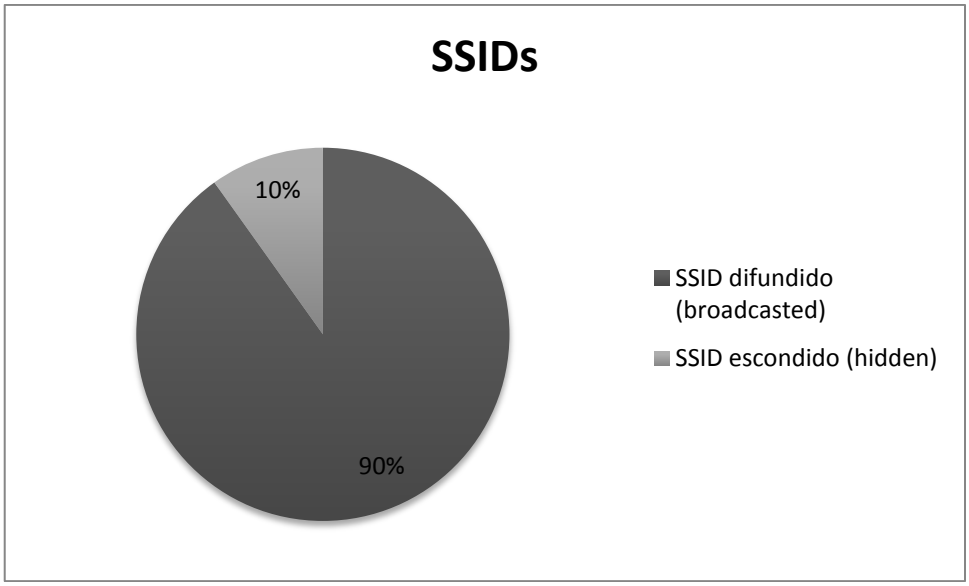

Figura 4.15 - SSIDs dos pontos de acesso localizados na cidade de Aveiro

A figura 4.15 apresenta-nos os SSIDs dos pontos de acesso localizados na cidade de Aveiro. De um total de 5478 pontos de acesso, 541 (10%) apresentam um *hidden* SSID, os restantes 4937 difundem o seu SSID.

4.8 Sumário / Conclusões

Neste capítulo foi apresentado o ponto estrutural desta dissertação: o estudo e caracterização da infraestrutura disponível para uma rede não planeada. A cidade de Aveiro, escolhida como cidade alvo, apresenta uma estrutura bastante consistente para conseguir construir o que se pretende num futuro próximo: uma rede sem fios em malha. Não obstante de todas as dificuldades encontradas, foi possível obter resultados práticos da cidade em estudo, alguns directamente, outros por estimativa. Estes resultados apresentam-se importantes, não só para a rede sem fios em malha em projecção, mas também para fornecimento de dados estatísticos da própria cidade.

Capítulo 5

Conclusões

5.1 Conclusões gerais

Esta tese apresentava-se com um carácter bastante prático, tendo o objectivo de caracterizar a infraestrutura da rede encontrada na cidade de Aveiro. Este problema demonstrou-se um desafio interessante, onde foram sentidas bastantes dificuldades. Apesar de ser um tipo de estudo já efectuado em várias cidades, é novo nesta cidade, e portanto existem sempre ajustes que são necessários fazer tendo em conta aspectos específicos da própria cidade. Desta forma, e tendo como estímulo a necessidade deste tipo de estudos para o fornecimento de um serviço global e em grande escala, ao longo do tempo foi-se tendo em atenção vários pormenores para que o estudo fosse algo rigoroso.

Todas as decisões tomadas em todos os momentos foram descritas, detalhando assim num método único e aplicado exclusivamente a esta cidade. A área de estudo era grande, o que para o mapeamento através de *warbiking*, levou a um processo cansativo e de alguma demora.

Para todo o trabalho foi sempre necessário recorrer a várias pesquisas e inclusive a um trabalho prático, de forma a interiorizar melhor conceitos, que apenas lidos não se tornam tão perceptíveis. Contudo, os conceitos teóricos foram sempre mantidos comparativamente com a prática, para que houvesse uma linha de seguimento.

Apresentando todos os resultados obtidos, isto é, caracterizando a topologia da rede mapeada na cidade de Aveiro é agora possível concluir sobre a viabilização de construção de uma rede sem fios em malha sobre esta topologia. A necessidade da simulação é um facto importante, e que tem peso nas conclusões do projecto. A ausência de uma simulação mais próxima da realidade

é um facto que faz com que não haja uma aprovação tão rápida dos valores obtidos. Tendo em conta todos os resultados obtidos, numa primeira análise, verifica-se que apesar do número de nós se apresentar num bom número, 5478, o número de habitantes é bastante superior ao número de nós (cerca de quatro vezes mais). Este valor talvez não fosse preocupante se existisse homogeneidade na distribuição de nós pela cidade, mas não é o caso. Isso aponta-nos para uma densidade excessiva em algumas zonas, e uma baixa densidade de terminais noutras. Este facto reflecte-se no número de ilhas encontradas. Para referência, o alcance de 50 metros é o *standard*. Neste caso encontraram-se 60 ilhas, o que para uma cidade com 12Km^2 e 5478 nós, apresenta-se com bastantes lacunas, necessitando assim de bastantes *gateways* para o efeito desejado. Este estudo apresenta-se como um bom marco neste processo de viabilização, no entanto é necessário dispor mais tempo com a simulação, para que se possam obter resultados definitivos. A meu ver, é possível a concretização de uma rede sem fios em malha na cidade de Aveiro, mas seria necessário introduzir alguns nós extra, de forma a que a densidade de terminais fosse homogeneizada pela cidade. Cumprindo este requesito, seria possível ajustar o alcance necessário para obter uma conectividade considerável, e depois desta forma, seria possível obter valores para largura de banda disponível para um cliente num dado nó, através de simulação directa e não de uma estimativa grosseira, o que nos levaria a uma aproximação mais real.

5.2 Contribuições

Esta dissertação apresentou-se pioneira neste campo na cidade de Aveiro. Nenhum estudo público até agora é conhecido. Recorrendo aos dados obtidos directamente da recolha de dados, outros por tratamento dos dados e posterior simulação e com ajuda de indicadores estatísticos genéricos foi possível construir uma base de resultados e dados que não existiam referentes a esta cidade. Esta dissertação apresenta-se assim como o princípio de um processo em estudo. Será possível agora com os dados obtidos, obter novos e com isso concluir sobre novos aspectos.

Apesar de o resultado final ser ainda inconclusivo, todo o esforço e dedicação não foi em vão, pois com tudo que foi feito, é possível verificar o que foi mal pensado e estruturado, refazer e obter novos resultados para novos estudos.

5.3 Trabalho futuro

A realização da construção da infra-estrutura encontra-se ainda a alguns passos de distância após este estudo. Como este trabalho é algo limitado e apenas estimando valores, não é possível concluir sobre vários factores. Um destes factores é o factor de adesão; era necessário sondar o público alvo desta rede e lidar com números concretos para obter noção de custos, e desta forma verificar se compensaria uma rede deste tipo. Pensar se seria um serviço pago ou grátis é também um trabalho futuro, pois sendo pago, existem necessidades às quais nunca se podem falhar, necessitando assim de uma estrutura bem idealizada. Assim, um papel importante a desenvolver após esta primeira análise seria, através de todos os dados obtidos nesta dissertação, fazer uma relação alcance custo, de forma a verificar qual a melhor forma de obter bom desempenho na rede.

Tendo em conta que seria possível viabilizar uma rede através desta topologia, talvez fosse necessário ajustar a mesma, para que se pudesse atingir os valores estimados ou melhores sem inclusão de custos.

Finalmente, mais tarde, tendo sido testada nesta área de estudo, referentes ao centro da cidade de Aveiro, talvez fosse possível expandir todo este trabalho ao concelho, visto que o centro da cidade apenas representa 6% do concelho, e uma rede desta envergadura representaria uma enorme evolução tecnológica em Portugal.

Referências

[1] Site Oficial WiFi Tastic. [Online]. http://www.wifitastic.com/

[2] Site Oficial FON. [Online]. http://www.fon.com

[3] J.P. Shim, Seungjae Shin, and M.B.H. Weiss, "Wireless internet competition: Municipal wireless vs. 3G mobile service," in *Wireless Telecommunications Symposium*, 2007, pp. 1-6.

[4] G. Valadon, F. Le Goff, and C. Berger, "A Practical Characterization of 802.11 Access Points in Paris," in *Fifth Advanced International Conference on Telecommunications*, 2009, pp. 220-225.

[5] K. Jones and Ling Liu, "What Where Wi: An Analysis of Millions of Wi-Fi Access Points," in *Portable Information Devices*, 2007, pp. 1-4.

[6] Site Oficial Instituto de Telecomunicações. [Online]. http://www.it.pt/

[7] Site Oficial IEEE. [Online]. http://www.ieee.org

[8] Site Oficial Wi-Fi Alliance. [Online]. http://www.wifi.org

[9] IEEE Std 802.11n, , 2009.

[10] David J. Thuente, Benjamin Newlin, and Mithun Acharya, "Jamming Vulnerabilities of IEEE 802.11e," in *Military Communications Conference*, 2007, pp. 1-7.

[11] S.-C Wang, Y.-M. Chen, Tsern-Huei Lee, and A Helmy, "Performance Evaluations for Hybrid IEEE 802.11b and 802.11g Wireless Networks," in *Performance, Computing and Communication Conference*, 2005, pp. 111-118.

[12] Yong Bai, Yifan Yu, and Lan Chen, "Enhanced Protection Mechanism for Improving Co-existence of IEEE 802.11b and IEEE 802g Wireless LANs," in *Vehicular Technology Conference*, 2009, pp. 1-5.

[13] S. Kapp, "802.11a. More bandwidth without the wires," *IET Journal* , vol. 6, no. Internet Computing, IEEE, pp. 75-79, Agosto 2002.

[14] Site do standard IEEE 802.11. [Online]. http://standards.ieee.org/getieee802/

[15] G.R. Hiertz, S. Max, T. Junge, D. Denteneert, and L. Berlemann, "IEEE 802.11s - Mesh Deterministic Access," in *Wireless Conference* , 2008, pp. 22-25.

[16] G.R. Hiertz, S. Max, Rui Zhao, D. Denteneer, and L. Berlemann, "Principles of IEEE 802.11s," in *Computer Communications and Networks*, 2007, pp. 1002-1007.

[17] S. Kawade, T.G. Hodgkinson, and V. Abhayawardhana, "Interference Analysis of 802.11b and 802.11g Wireless Systems," in *Vehicular Technology Conference*, 2007, pp. 787-791.

[18] S. Kawade and T.G. Hodgkinson, "Analysis of Interference Effects between Co-Existent 802.11b and 802.11g Wi-Fi Systems," in *VehicularTechnology Conference*, 2008, pp. 1881-1885.

[19] J.P. Hauser and D.J. Baker, "Mobility and routing protocols for 802.11 extended service sets," in *Military Communications Conference*, 2003, pp. 13-16, Vol. 2.

[20] S. Narayanan, Pei Liu, and S.S. Panwar, "On the advantages of multi-hop extensions to the IEEE 802.11 infrastructure mode," in *Wireless Communications and Networking Conference*, 2005, pp. 132-138, vol.2.

[21] R. Schmitz, M. Torrent-Moreno, H. Hartenstein, and W. Effelsberg, "The impact of wireless radio fluctuations on ad hoc network performance," in *Local Computer Networks*, 2004, pp. 594-601.

[22] Tim Baugé. (2004, Maio) Ad hoc networking in military scenarios. [Online]. http://www.thalesgroup.com

[23] G. Zussman and A. Segall, "Energy efficient routing in ad hoc disaster recovery networks," in *INFOCOM*, 2003, pp. 682-691, vol. 1.

[24] P. Fallara, "Disaster recovery planning," *IET Journal or Magazine*, vol. 22, pp. 42-44, 2003.

[25] N. Aschenbruck, C. de Waal, and P. Martini, "Distribution of nodes in disaster area scenarios and its impact on topology control strategies," in *Computer Communications Workshops*, 2008, pp. 1-6.

[26] F. Kojima and M. Fujise, "Feasibility Study on VHF Band Based Ad-Hoc Disaster Prevention Radio," in *ITS Telecommunications Proceedings*, 2006, pp. 663-666.

[27] D. Asahizawa, Y. Sato, G. Sato, and Y. Shibata, "Disaster Surveillance Video Transmission System by Wireless Ballooned Network," in *Advanced Information Networking and Applications Workshops*, 2009, pp. 884-889.

[28] Y. Shibata, Y. Sato, N. Ogasawara, and G. Chiba, "A Disaster Information System by Ballooned Wireless Adhoc Network," in *Complex, Intelligent and Software Intensive Systems*, 2009, pp. 299-304.

[29] Site Oficial USI Wireless Minneapolis. [Online]. http://www.usiwireless.com

[30] Site Oficial Wireless Minneapolis. [Online]. http://www.ci.minneapolis.mn.us/wirelessminneapolis/

[31] Site Oficial Google Wifi. [Online]. http://wifi.google.com

[32] Site Oficial "One laptop per child". [Online]. http://laptop.org/en/vision/index.shtml

[33] T. Braun, "Wireless Mesh Networks for Meteorological Monitoring," in *Distributed Computing Systems Workshops*, 2009, p. 425.

[34] L. Berlemann, C. Hoymann, G.R. Hiertz, and S. Mangold, "Coexistence and Interworking of IEEE 802.16 and IEEE 802.11(e)," in *Vehicular Technology Conference*, 2006, pp. 7-10, Vol. 1.

[35] Site Oficial Netkrom. [Online]. http://www.netkrom.com/

[36] Site Oficial MetroMesh. [Online]. http://www.metromesh.com.au/

[37] N. Ghazisaidi, H. Kassaei, and M.S. Bohlooli, "Integration of WiFi and WiMAX-Mesh Networks," in *Advances in Mesh Networks*, 2009, pp. 1-6.

[38] Site Oficial Task Group S. [Online]. http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm

[39] Site Oficial Open 802.11. [Online]. http://www.open80211s.org

[40] Site Oficial MITRE. [Online]. http://www.mitre.org/

[41] D.J. Shyy, "Military Usage Scenario and IEEE 802.11s Mesh Networking Standard," in *Military Communications Conference*, 2006, pp. 23-25.

[42] Site Oficial Ubuntu. [Online]. http://www.ubuntu.com/

[43] Site Oficial Ferramenta IPerf. [Online]. http://www.noc.ucf.edu/Tools/Iperf

[44] Site Oficial NetStumbler. [Online]. http://www.netstumbler.com

[45] Site Oficial Chanalyzer. [Online]. http://www.metageek.net/products/chanalyzer-3

[46] Site Oficial Putty. [Online]. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[47] Site Oficial Câmara Municipal de Aveiro. [Online]. www.cm-aveiro.pt

[48] Minkyong Kim, Jeffrey J. Fielding, and David Kotz, "Risks of using AP locations discovered through war driving," Department of Computer Science Dartmouth College, 2006.

[49] Arvin Wen Tsui, Wei-Cheng Lin, and Hao-hua Chu, "Analysis and Comparison between War Driving and War Walking in Metropolitan WiFi Radio Maps," Department of Computer Science and Information Engineering, National Taiwan University; Information and Communication Research Lab., Industrial Technology Research Institute, Taiwan,.

[50] Site Ubiquiti SRX. [Online]. http://www.ubnt.com/products/srx.php

[51] Site Oficial Kismet. [Online]. http://www.kismetwireless.net/

[52] Site WBT-201. [Online]. http://www.wintec.com.tw/en/product_detail.php?pro_id=65

[53] Site Garmin Nuvi 300. [Online]. https://buy.garmin.com/shop/shop.do?pID=339&ra=true#specsTab

[54] Site Explicativo Urban Canyon Effect. [Online]. http://en.wikipedia.org/wiki/Urban_canyon

[55] Site Fórmula Vicenty. [Online]. http://www.movable-type.co.uk/scripts/latlong-vincenty.html

[56] Site Oficial NS-2. [Online]. http://www.isi.edu/nsnam/ns/

[57] E. Weingartner, H. vom Lehn, and K. Wehrle, "A Performance Comparison of Recent Network Simulators," in *Communications*, 2009, pp. 1-5.

[58] Site Oficial NS-3. [Online]. http://www.nsnam.org/

[59] Site Oficial Livraria Boost Graph. [Online]. http://www.boost.org/doc/libs/1_40_0/libs/graph/doc/index.html

[60] Sputnik, "RF Propagation Basics," , 2004, http://www.sputnik.com/docs/rf_propagation_basics.pdf.

[61] J. del Prado Pavon and S.N. Shankar, "Impact of frame size, number of stations and mobility on the throughput performance of IEEE 802.11e," in *Wireless Communications and Networking Conference*, 2004, pp. 21-25, Vol. 2.

[62] J. Lorincz and D. Begusic, "Physical layer analysis of emerging IEEE 802.11n WLAN standard," in *Advanced Communication Technology*, 2006, pp. 20-22, Vol. 1.

[63] José L. F. dos Reis, Filipe Lopes, Pedro Miguens, and Pedro Sampaio, "Análise e comparação de algoritmos de cálculo de posições GPS," Departamento de Engenharia Electrónica e Telecomunicações e de Computadores (DEETC) - ISEL, Lisboa,.

[64] Ken Fong et al., "Wireless LAN War Driving Survey," Professional Information Security Association (PISA); Hong Kong Wireless Technology Industry Association (WTIA), Hong Kong, 2006.

[65] Y. Shibata, Y. Sato, N. Ogasawara, and G. Chiba, "Network, A Disaster Information System by Ballooned Wireless Adhoc," in *Complex, Intelligent and Software Intensive Systems*, 2009, pp. 299-304.

[66] Site Oficial Task Group N IEEE. [Online]. http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm

Anexos

I – Configuração do Kismet (kismet.conf)

```
# Kismet config file
# Most of the "static" configs have been moved to here -- the command
line
# config was getting way too crowded and cryptic.  We want functionality,
# not continually reading --help!

# Version of Kismet config
version=2009-newcore

# Name of server (Purely for organizational purposes)
servername=Kismet_2009

# Prefix of where we log (as used in the logtemplate later)
# logprefix=/some/path/to/logs

# Do we allow plugins to be used?
allowplugins=true

# See the README for full information on the new source format
# ncsource=interface:options
# for example:
# ncsource=wlan0
# ncsource=wifi0:type=madwifi
# ncsource=wlan0:name=intel,hop=false,channel=11
ncsource=wlan1:name=ath5k,forcevap=false

# Comma-separated list of sources to enable.  This is only needed if you
defined
# multiple sources and only want to enable some of them.  By default, all
defined
# sources are enabled.
# For example:
# enablesources=prismsource,ciscosource

# How many channels per second do we hop?  (1-10)
channelvelocity=5

# By setting the dwell time for channel hopping we override the
channelvelocity
# setting above and dwell on each channel for the given number of
seconds.
#channeldwell=10

# Channels are defined as:
# channellist=name:ch1,ch2,ch3
# or
# channellist=name:range-start-end-width-offset,ch,range,ch,...
#
# Channels may be a numeric channel or a frequency
#
# Channels may specify an additional wait period.  For common default
channels,
# an additional wait period can be useful.  Wait periods delay for that
number
# of times per second - so a configuration hopping 10 times per second
with a
```

```
# channel of 6:3 would delay 3/10ths of a second on channel 6.
#
# Channel lists may have up to 256 channels and ranges (combined).  For
power
# users scanning more than 256 channels with a single card, ranges must
be used.
#
# Ranges are meant for "power users" who wish to define a very large
number of
# channels.  A range may specify channels or frequencies, and will
automatically
# sort themselves to cover channels in a non-overlapping fashion.  An
example
# range for the normal 802.11b/g spectrum would be:
#
# range-1-11-3-1
#
# which indicates starting at 1, ending at 11, a channel width of 3
channels,
# incrementing by one.  A frequency based definition would be:
#
# range-2412-2462-22-5
#
# since 11g channels are 22 mhz wide and 5 mhz apart.
#
# Ranges have the flaw that they cannot be shared between sources in a
non-overlapping
# way, so multiple sources using the same range may hop in lockstep with
each other
# and duplicate the coverage.
#
# channellist=demo:1:3,6:3,11:3,range-5000-6000-20-10

# Default channel lists
# These channel lists MUST BE PRESENT for Kismet to work properly.  While
it is
# possible to change these, it is not recommended.  These are used when
the supported
# channel list can not be found for the source; to force using these
instead of
# the detected supported channels, override with channellist= in the
source defintion
channellist=IEEE80211b:1:3,6:3,11:3,2,7,3,8,4,9,5,10
channellist=IEEE80211a:36,40,44,48,52,56,60,64,149,153,157,161,165
channellist=IEEE80211ab:1:3,6:3,11:3,2,7,3,8,4,9,5,10,36,40,44,48,52,56,6
0,64,149,153,157,161,165

# Client/server listen config
listen=tcp://127.0.0.1:2501
# People allowed to connect, comma seperated IP addresses or network/mask
# blocks.  Netmasks can be expressed as dotted quad (/255.255.255.0) or
as
# numbers (/24)
allowedhosts=127.0.0.1
# Maximum number of concurrent GUI's
maxclients=5
# Maximum backlog before we start throwing out or killing clients.  The
# bigger this number, the more memory and the more power it will use.
maxbacklog=5000
```

```
# Server + Drone config options
# dronelisten=tcp://127.0.0.1:3501
# droneallowedhosts=127.0.0.1
# dronemaxclients=5
# droneringlen=65535

# OUI file, expected format 00:11:22<tab>manufname
# IEEE OUI file used to look up manufacturer info.  We default to the
# wireshark one since most people have that.
ouifile=/etc/manuf
ouifile=/usr/share/wireshark/wireshark/manuf
ouifile=/usr/share/wireshark/manuf

# Do we have a GPS?
gps=true
# Do we use a locally serial attached GPS, or use a gpsd server?
# (Pick only one)
gpstype=gpsd
# gpstype=serial
# What serial device do we look for the GPS on?
#gpsdevice=/dev/rfcomm0
gpsdevice=/dev/ttyUSB0
# Host:port that GPSD is running on.  This can be localhost OR remote!
gpshost=localhost:2947
# Do we lock the mode?  This overrides coordinates of lock "0", which
will
# generate some bad information until you get a GPS lock, but it will
# fix problems with GPS units with broken NMEA that report lock 0
gpsmodelock=false
# Do we try to reconnect if we lose our link to the GPS, or do we just
# let it die and be disabled?
gpsreconnect=true

# Do we export packets over tun/tap virtual interfaces?
tuntap_export=false
# What virtual interface do we use
tuntap_device=kistap0

# Packet filtering options:
# filter_tracker - Packets filtered from the tracker are not processed or
#                  recorded in any way.
# filter_dump    - Packets filtered at the dump level are tracked,
displayed,
#                  and written to the csv/xml/network/etc files, but not
#                  recorded in the packet dump
# filter_export  - Controls what packets influence the exported CSV,
network,
#                  xml, gps, etc files.
# All filtering options take arguments containing the type of address and
# addresses to be filtered.  Valid address types are 'ANY', 'BSSID',
# 'SOURCE', and 'DEST'.  Filtering can be inverted by the use of '!'
before
# the address.  For example,
# filter_tracker=ANY(!"00:00:DE:AD:BE:EF")
# has the same effect as the previous mac_filter config file option.
# filter_tracker=...
# filter_dump=...
# filter_export=...
```

```
# filter_netclient=...

# Alerts to be reported and the throttling rates.
# alert=name,throttle/unit,burst
# The throttle/unit describes the number of alerts of this type that are
# sent per time unit.  Valid time units are second, minute, hour, and
day.
# Burst describes the number of alerts sent before throttling takes
place.
# For example:
# alert=FOO,10/min,5
# Would allow 5 alerts through before throttling is enabled, and will
then
# limit the number of alerts to 10 per minute.
# A throttle rate of 0 disables throttling of the alert.
# See the README for a list of alert types.
alert=ADHOCCONFLICT,5/min,1/sec
alert=AIRJACKSSID,5/min,1/sec
alert=APSPOOF,10/min,1/sec
alert=BCASTDISCON,5/min,2/sec
alert=BSSTIMESTAMP,5/min,1/sec
alert=CHANCHANGE,5/min,1/sec
alert=CRYPTODROP,5/min,1/sec
alert=DISASSOCTRAFFIC,10/min,1/sec
alert=DEAUTHFLOOD,5/min,2/sec
alert=DEAUTHCODEINVALID,5/min,1/sec
alert=DISCONCODEINVALID,5/min,1/sec
alert=DHCPNAMECHANGE,5/min,1/sec
alert=DHCPOSCHANGE,5/min,1/sec
alert=DHCPCLIENTID,5/min,1/sec
alert=DHCPCONFLICT,10/min,1/sec
alert=NETSTUMBLER,5/min,1/sec
alert=LUCENTTEST,5/min,1/sec
alert=LONGSSID,5/min,1/sec
alert=MSFBCOMSSID,5/min,1/sec
alert=MSFDLINKRATE,5/min,1/sec
alert=MSFNETGEARBEACON,5/min,1/sec
alert=NULLPROBERESP,5/min,1/sec
#alert=PROBENOJOIN,5/min,1/sec

# Controls behavior of the APSPOOF alert.  SSID may be a literal match
(ssid=) or
# a regex (ssidregex=) if PCRE was available when kismet was built.  The
allowed
# MAC list must be comma-separated and enclosed in quotes if there are
multiple
# MAC addresses allowed.  MAC address masks are allowed.
apspoof=Foo1:ssidregex="(?i:foobar)",validmacs=00:11:22:33:44:55
apspoof=Foo2:ssid="Foobar",validmacs="00:11:22:33:44:55,aa:bb:cc:dd:ee:ff
"

# Known WEP keys to decrypt, bssid,hexkey.  This is only for networks
where
# the keys are already known, and it may impact throughput on slower
hardware.
# Multiple wepkey lines may be used for multiple BSSIDs.
# wepkey=00:DE:AD:C0:DE:00,FEEDFACEDEADBEEF0102030405060708090
```

```
# Is transmission of the keys to the client allowed?  This may be a
security
# risk for some.  If you disable this, you will not be able to query keys
from
# a client.
allowkeytransmit=true

# How often (in seconds) do we write all our data files (0 to disable)
writeinterval=200

# Do we use sound?
# Not to be confused with GUI sound parameter, this controls wether or
not the
# server itself will play sound.  Primarily for headless or automated
systems.
enablesound=false
# Path to sound player
soundbin=play

sound=newnet,true
sound=newcryptnet,true
sound=packet,true
sound=gpslock,true
sound=gpslost,true
sound=alert,true

# Does the server have speech? (Again, not to be confused with the GUI's
speech)
enablespeech=false
# Binary used for speech (if not in path, full path must be specified)
speechbin=flite
# Specify raw or festival; Flite (and anything else that doesn't need
formatting
# around the string to speak) is 'raw', festival requires the string be
wrapped in
# SayText("...")
speechtype=raw

# How do we speak?  Valid options:
# speech    Normal speech
# nato      NATO spellings (alpha, bravo, charlie)
# spell     Spell the letters out (aye, bee, sea)
speechencoding=nato

speech=new,"New network detected s.s.i.d. %1 channel %2"
speech=alert,"Alert %1"
speech=gpslost,"G.P.S. signal lost"
speech=gpslock,"G.P.S. signal O.K."

# How many alerts do we backlog for new clients?  Only change this if you
have
# a -very- low memory system and need those extra bytes, or if you have a
high
# memory system and a huge number of alert conditions.
alertbacklog=50

# File types to log, comma seperated.  Built-in log file types:
# alert                         Text file of alerts
# gpsxml                        XML per-packet GPS log
```

```
# nettxt                        Networks in text format
# netxml                        Networks in XML format
# pcapdump                      tcpdump/wireshark compatible pcap log file
# string                        All strings seen (increases CPU load)
logtypes=alert,pcapdump,gpsxml,netxml,nettxt

# Format of the pcap dump (PPI or 80211)
pcapdumpformat=ppi
#pcapdumpformat=80211

# Default log title
logdefault=kismet

# logtemplate - Filename logging template.
# This is, at first glance, really nasty and ugly, but you'll hardly ever
# have to touch it so don't complain too much.
#
# %p is replaced by the logging prefix + '/'
# %n is replaced by the logging instance name
# %d is replaced by the starting date as Mon-DD-YYYY
# %D is replaced by the current date as YYYYMMDD
# %t is replaced by the starting time as HH-MM-SS
# %i is replaced by the increment log in the case of multiple logs
# %l is replaced by the log type (pcapdump, strings, etc)
# %h is replaced by the home directory

logtemplate=%p%n-%i.%l

# Where state info, etc, is stored.  You shouldnt ever need to change
this.
# This is a directory.
configdir=%h/.kismet/
```

II – *Script* gps2ns2.py

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009"
SINTAXE:    script.py [ficheiro entrada] [ficheiro saida (opcional)]
      [ficheiro entrada]      Apenas converte ficheiros *.gpsxml

OUTPUT: [ficheiro saida]
      Por omissão: [ficheiro_entrada].ns2
"""

#Função que calcula a distância em metros entre dois pontos
#com coordenadas usando a formula inversa de Vincenty para elipsoides
def vinc_dist(  f,  a,  phi1,  lembda1,  phi2,  lembda2 ) :
        """

        Returns the distance between two geographic points on the
ellipsoid
        and the forward and reverse azimuths between these points.
        lats, longs and azimuths are in radians, distance in metres

        Returns ( s, alpha12,  alpha21 ) as a tuple

        """

        if (abs( phi2 - phi1 ) < 1e-8) and ( abs( lembda2 - lembda1) <
1e-8 ) :
          return 0.0, 0.0, 0.0

        two_pi = 2.0*math.pi

        b = a * (1.0 - f)

        TanU1 = (1-f) * math.tan( phi1 )
        TanU2 = (1-f) * math.tan( phi2 )

        U1 = math.atan(TanU1)
        U2 = math.atan(TanU2)

        lembda = lembda2 - lembda1
        last_lembda = -4000000.0                # an impossibe value
        omega = lembda

        # Iterate the following equations,
        #  until there is no significant change in lembda

        while ( last_lembda < -3000000.0 or lembda != 0 and abs(
(last_lembda - lembda)/lembda) > 1.0e-9 ) :

          sqr_sin_sigma = pow( math.cos(U2) * math.sin(lembda), 2) + \
                pow( (math.cos(U1) * math.sin(U2) - \
                math.sin(U1) *  math.cos(U2) * math.cos(lembda) ), 2 )
```

```
            Sin_sigma = math.sqrt( sqr_sin_sigma )

            Cos_sigma = math.sin(U1) * math.sin(U2) + math.cos(U1) *
math.cos(U2) * math.cos(lembda)

            sigma = math.atan2( Sin_sigma, Cos_sigma )

            Sin_alpha = math.cos(U1) * math.cos(U2) * math.sin(lembda) /
math.sin(sigma)
            alpha = math.asin( Sin_alpha )

            Cos2sigma_m = math.cos(sigma) - (2 * math.sin(U1) *
math.sin(U2) / pow(math.cos(alpha), 2) )

            C = (f/16) * pow(math.cos(alpha), 2) * (4 + f * (4 - 3 *
pow(math.cos(alpha), 2)))

            last_lembda = lembda

            lembda = omega + (1-C) * f * math.sin(alpha) * (sigma + C *
math.sin(sigma) * \
                  (Cos2sigma_m + C * math.cos(sigma) * (-1 + 2 *
pow(Cos2sigma_m, 2) )))


        u2 = pow(math.cos(alpha),2) * (a*a-b*b) / (b*b)

        A = 1 + (u2/16384) * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))

        B = (u2/1024) * (256 + u2 * (-128+ u2 * (74 - 47 * u2)))

        delta_sigma = B * Sin_sigma * (Cos2sigma_m + (B/4) * \
                (Cos_sigma * (-1 + 2 * pow(Cos2sigma_m, 2) ) - \
                (B/6) * Cos2sigma_m * (-3 + 4 * sqr_sin_sigma) * \
                (-3 + 4 * pow(Cos2sigma_m,2 ) )))

        s = b * A * (sigma - delta_sigma)

        alpha12 = math.atan2( (math.cos(U2) * math.sin(lembda)), \
                (math.cos(U1) * math.sin(U2) - math.sin(U1) *
math.cos(U2) * math.cos(lembda)))

        alpha21 = math.atan2( (math.cos(U1) * math.sin(lembda)), \
                (-math.sin(U1) * math.cos(U2) + math.cos(U1) *
math.sin(U2) * math.cos(lembda)))

        if ( alpha12 < 0.0 ) :
                alpha12 =  alpha12 + two_pi
        if ( alpha12 > two_pi ) :
                alpha12 = alpha12 - two_pi

        alpha21 = alpha21 + two_pi / 2.0
        if ( alpha21 < 0.0 ) :
                alpha21 = alpha21 + two_pi
        if ( alpha21 > two_pi ) :
                alpha21 = alpha21 - two_pi

        return s, alpha12,  alpha21
```

```
#Função que extrai toda a informação do ficheiro *.gps
#lê a i linha do ficheiro *.gps em formato xml
#Input:linha, ficheiro
#Output:end (sinal de controlo), bssid, source, lat, lon, signal
def xml_parser(i,file_input):
      try:
            gps_run = xml_file.getroot()
            #Output
            end = 0
            bssid = gps_run[i].get("bssid")
            source = gps_run[i].get("source")
            lat = gps_run[i].get("lat")
            lon = gps_run[i].get("lon")
            signal = gps_run[i].get("signal_dbm")
      except Exception, inst:
            #Output
            end = 1
            bssid = -999
            source = -999
            lat = 999
            lon = 999
            signal = -999
      return      end, bssid, source, lat, lon, signal

#Função que devolve o ponto com maior latitude e menor longitude
#Input: ficheiro
#Output: lat_min, lon_min
def corner_values(file_input_name):
      lat_min = 999
      lon_min = 999
      i=1
      while(1):
            res = xml_parser(i,file_input_name)
            if res[0] ==1:
                  break
            lat = float(res[3])
            lon = float(res[4])
            #exclui linhas do gps e linhas dos clientes ligados aos aps
            if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG" and
res[1] != "00:00:00:00:00:00"):
                  #procura pelo ponto de menor latitude e menor longitude
                  #NOTA: esta regra aplica-se a locais com latitudes
decimais positivas
                  # e longitudes decimais negativas, neste caso
específico: Aveiro
                  if lat < lat_min:
                        lat_min=lat
                  if lon < lon_min:
                        lon_min=lon
            i+=1
      return lat_min, lon_min

#Função que gera duas listas para filtrar a informação extraida do
ficheiro *.gps
#Input: ficheiro
#Output: keys (lista com os macs dos aps), info (lista com todos os ponto
capturados dos aps)
def create_lists(file_input_name):
```

```
	keys = []
	info = []
	i=1
	j=0
	while(1):
			res = xml_parser(i,file_input_name)
			if res[0] ==1:
				break
			bssid = str(res[1])
			#exclui linhas do gps e linhas dos clientes ligados aos
aps
			if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG"
and res[1] != "00:00:00:00:00:00"):
				lat = res[3]
				lon = res[4]
				signal = res[5]
				if bssid in keys:
					index=keys.index(bssid)
					info[index].append([signal,lat,lon])
				else:
					keys.append(bssid)
					index=keys.index(bssid)
					info.insert(index,[[signal,lat,lon]])

			i+=1
	return keys, info


#Função que gera o ficheiro nodes.ns2
#Input: string, nome do ficheiro
#Output: ficheiro nodes.ns2
def ns2_file(nodes, file_output_name):
	try:
		file_output = open(file_output_name,'w')
		file_output.write ("#\n")
		file_output.write ("#nodes.ns2\n")
		file_output.write ("#\n")
		file_output.writelines (nodes)
		file_output.close()
		print "Ficheiro gerado!\n"
	except IOError:
		print "Erro ao criar o ficheiro: 'nodes.ns2'!\n"
		sys.exit(1)

#Função que devolve para cada ap a latitude e a longitude da amostra com
maior sinal
#Input: info (lista)
#Output: res (lista)
def best_signal(info):
	res = []
	for i in range(len(info)):
		res.append(info[i][0])
		for j in range(len(info[i])):
			if int(info[i][j][0]) < int(res[i][0]):
				res[i] = info[i][j]
	return res

args = sys.argv[1:]
if args:
```

```
	file_input_name=args[0]
	if (len(args)==2):
		file_output_name=args[1]+".ns2"
	else:
		file_output_name=file_input_name+".ns2"

	try:
		xml_file = ET.parse(file_input_name)
		print "\nFicheiro lido: %s!\n" %file_input_name
	except IOError:
		print "\nFicheiro '%s' NAO existe!\n" %file_input_name
		sys.exit(1)
	try:
		nodes = ""
		nodes_temp_x = ""
		nodes_temp_y = ""
		nodes_temp_z = ""
		bssid_= ""

		#procura o ponto (x,y,z) = (0,0,0)
		lat_min, lon_min =corner_values(file_input_name)

		#extrai a informação do gps e cria duas listas
		l=create_lists(file_input_name)
		#retorno da função create_lists
		keys = l[0]
		info = l[1]
		print "Numero total de nos encontrados: " + str(len(info)) +
"\n"

		#procura nos dados extraidos, para cada ap as coordenadas
		#	correspondentes ao maior sinal
		res = best_signal(info)


		#valores da elipsóide WGS-84
		a=6378137
		f=1/298.257223563

		i=0
		for i in range(len(res)):
			lat=float(res[i][1])
			lon=float(res[i][2])
			#calcula a distancia de dois pontos

	v=vinc_dist(f,a,math.radians(lat_min),math.radians(lon_min),math.ra
dians(lat),math.radians(lon))
			dist=v[0]
			angle = (math.pi/2) - v[1]
			x=dist*math.cos(angle)
			y= dist*math.sin(angle)
			nodes_temp_x='$node_(' + str(i) + ') set X_ ' + str(x)
+ "\n"
			nodes_temp_y='$node_(' + str(i) + ') set Y_ ' + str(y)
+ "\n"
			nodes_temp_z='$node_(' + str(i) + ') set Z_ 0.0\n'
			bssid_= "#SSID = " + keys[i] + "\n"
			nodes = nodes + bssid_ + nodes_temp_x + nodes_temp_y +
nodes_temp_z
```

```
                i+=1

            print "A gerar o ficheiro " + file_output_name + " ... \n"
            ns2_file(nodes, file_output_name)
      except IOError:
            print "Erro a gerar o ficheiro!"
            sys.exit(1)
else:
      Usage()
      sys.exit(1)
```

III – *Script* gps2ns3.py

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009"
SINTAXE:    script.py [ficheiro entrada] [ficheiro saida (opcional)]
      [ficheiro entrada]      Apenas converte ficheiros *.gpsxml

OUTPUT: [ficheiro saida]
      Por omissão: [ficheiro_entrada].ns3]
"""

#Função que calcula a distância em metros entre dois pontos
#com coordenadas usando a formula inversa de Vincenty para elipsoides
def vinc_dist(  f,  a,  phi1,  lembda1,  phi2,  lembda2 ) :
        """

        Returns the distance between two geographic points on the
ellipsoid
        and the forward and reverse azimuths between these points.
        lats, longs and azimuths are in radians, distance in metres

        Returns ( s, alpha12,  alpha21 ) as a tuple

        """

        if (abs( phi2 - phi1 ) < 1e-8) and ( abs( lembda2 - lembda1) <
1e-8 ) :
          return 0.0, 0.0, 0.0

        two_pi = 2.0*math.pi

        b = a * (1.0 - f)

        TanU1 = (1-f) * math.tan( phi1 )
        TanU2 = (1-f) * math.tan( phi2 )

        U1 = math.atan(TanU1)
        U2 = math.atan(TanU2)

        lembda = lembda2 - lembda1
        last_lembda = -4000000.0                # an impossibe value
        omega = lembda

        # Iterate the following equations,
        #  until there is no significant change in lembda

        while ( last_lembda < -3000000.0 or lembda != 0 and abs(
(last_lembda - lembda)/lembda) > 1.0e-9 ) :

          sqr_sin_sigma = pow( math.cos(U2) * math.sin(lembda), 2) + \
                pow( (math.cos(U1) * math.sin(U2) - \
                math.sin(U1) *  math.cos(U2) * math.cos(lembda) ), 2 )
```

```
          Sin_sigma = math.sqrt( sqr_sin_sigma )

          Cos_sigma = math.sin(U1) * math.sin(U2) + math.cos(U1) *
math.cos(U2) * math.cos(lembda)

          sigma = math.atan2( Sin_sigma, Cos_sigma )

          Sin_alpha = math.cos(U1) * math.cos(U2) * math.sin(lembda) /
math.sin(sigma)
          alpha = math.asin( Sin_alpha )

          Cos2sigma_m = math.cos(sigma) - (2 * math.sin(U1) *
math.sin(U2) / pow(math.cos(alpha), 2) )

          C = (f/16) * pow(math.cos(alpha), 2) * (4 + f * (4 - 3 *
pow(math.cos(alpha), 2)))

          last_lembda = lembda

          lembda = omega + (1-C) * f * math.sin(alpha) * (sigma + C *
math.sin(sigma) * \
                (Cos2sigma_m + C * math.cos(sigma) * (-1 + 2 *
pow(Cos2sigma_m, 2) )))


        u2 = pow(math.cos(alpha),2) * (a*a-b*b) / (b*b)

        A = 1 + (u2/16384) * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))

        B = (u2/1024) * (256 + u2 * (-128+ u2 * (74 - 47 * u2)))

        delta_sigma = B * Sin_sigma * (Cos2sigma_m + (B/4) * \
                (Cos_sigma * (-1 + 2 * pow(Cos2sigma_m, 2) ) - \
                (B/6) * Cos2sigma_m * (-3 + 4 * sqr_sin_sigma) * \
                (-3 + 4 * pow(Cos2sigma_m,2 ) )))

        s = b * A * (sigma - delta_sigma)

        alpha12 = math.atan2( (math.cos(U2) * math.sin(lembda)), \
                (math.cos(U1) * math.sin(U2) - math.sin(U1) *
math.cos(U2) * math.cos(lembda)))

        alpha21 = math.atan2( (math.cos(U1) * math.sin(lembda)), \
                (-math.sin(U1) * math.cos(U2) + math.cos(U1) *
math.sin(U2) * math.cos(lembda)))

        if ( alpha12 < 0.0 ) :
                alpha12 =  alpha12 + two_pi
        if ( alpha12 > two_pi ) :
                alpha12 = alpha12 - two_pi

        alpha21 = alpha21 + two_pi / 2.0
        if ( alpha21 < 0.0 ) :
                alpha21 = alpha21 + two_pi
        if ( alpha21 > two_pi ) :
                alpha21 = alpha21 - two_pi

        return s, alpha12,  alpha21
```

```
#Função que extrai toda a informação do ficheiro *.gps
#lê a i linha do ficheiro *.gps em formato xml
#Input:linha, ficheiro
#Output:end (sinal de controlo), bssid, source, lat, lon, signal
def xml_parser(i,file_input):
      try:
            gps_run = xml_file.getroot()
            #Output
            end = 0
            bssid = gps_run[i].get("bssid")
            source = gps_run[i].get("source")
            lat = gps_run[i].get("lat")
            lon = gps_run[i].get("lon")
            signal = gps_run[i].get("signal_dbm")
      except Exception, inst:
            #Output
            end = 1
            bssid = -999
            source = -999
            lat = 999
            lon = 999
            signal = -999
      return      end, bssid, source, lat, lon, signal

#Função que devolve o ponto com maior latitude e menor longitude
#Input: ficheiro
#Output: lat_min, lon_min
def corner_values(file_input_name):
      lat_min = 999
      lon_min = 999
      i=1
      while(1):
            res = xml_parser(i,file_input_name)
            if res[0] ==1:
                  break
            lat = float(res[3])
            lon = float(res[4])
            #exclui linhas do gps e linhas dos clientes ligados aos aps
            if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG" and
res[1] != "00:00:00:00:00:00"):
                  #procura pelo ponto de menor latitude e menor longitude
                  #NOTA: esta regra aplica-se a locais com latitudes
decimais positivas
                  # e longitudes decimais negativas, neste caso
específico: Aveiro
                  if lat < lat_min:
                        lat_min=lat
                  if lon < lon_min:
                        lon_min=lon
            i+=1
      return lat_min, lon_min

#Função que gera duas listas para filtrar a informação extraida do
ficheiro *.gps
#Input: ficheiro
#Output: keys (lista com os macs dos aps), info (lista com todos os ponto
capturados dos aps)
def create_lists(file_input_name):
```

```
	keys = []
	info = []
	i=1
	j=0
	while(1):
			res = xml_parser(i,file_input_name)
			if res[0] ==1:
				break
			bssid = str(res[1])
			#exclui linhas do gps e linhas dos clientes ligados aos
aps
			if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG"
and res[1] != "00:00:00:00:00:00"):
				lat = res[3]
				lon = res[4]
				signal = res[5]
				if bssid in keys:
					index=keys.index(bssid)
					info[index].append([signal,lat,lon])
				else:
					keys.append(bssid)
					index=keys.index(bssid)
					info.insert(index,[[signal,lat,lon]])

			i+=1
	return keys, info


#Função que gera o ficheiro nodes.ns2
#Input: string, nome do ficheiro
#Output: ficheiro nodes.ns3
def ns3_file(nodes, file_output_name):
	try:
		file_output = open(file_output_name,'w')
		#file_output.write ("#\n")
		#file_output.write ("#nodes.ns3\n")
		#file_output.write ("#\n")
		file_output.writelines (nodes)
		file_output.close()
		print "Ficheiro gerado!\n"
	except IOError:
		print "Erro ao criar o ficheiro: '" + file_output_name +
"'!\n"
		sys.exit(1)

#Função que devolve para cada ap a latitude e a longitude da amostra com
maior sinal
#Input: info (lista)
#Output: res (lista)
def best_signal(info):
	res = []
	for i in range(len(info)):
		res.append(info[i][0])
		for j in range(len(info[i])):
			if int(info[i][j][0]) < int(res[i][0]):
				res[i] = info[i][j]
	return res

args = sys.argv[1:]
```

```
if args:
      file_input_name=args[0]
      if (len(args)==2):
            file_output_name=args[1]+".ns3"
      else:
            file_output_name=file_input_name+".ns3"

      try:
            xml_file = ET.parse(file_input_name)
            print "\nFicheiro lido: %s!\n" %file_input_name
      except IOError:
            print "\nFicheiro '%s' NAO existe!\n" %file_input_name
            sys.exit(1)
      try:
            nodes = ""
            #nodes_temp_x = ""
            #nodes_temp_y = ""
            #nodes_temp_z = ""
            nodes_temp_x_y_z = ""
            bssid_= ""
            number = 0

            #procura o ponto (x,y,z) = (0,0,0)
            lat_min, lon_min =corner_values(file_input_name)

            #extrai a informação do gps e cria duas listas
            l=create_lists(file_input_name)
            #retorno da função create_lists
            keys = l[0]
            info = l[1]
            print "Numero total de nos encontrados: " + str(len(info)) +
"\n"

            #procura nos dados extraidos, para cada ap as coordenadas
            #     correspondentes ao maior sinal
            res = best_signal(info)


            #valores da elipsóide WGS-84
            a=6378137
            f=1/298.257223563

            i=0
            for i in range(len(res)):
                  lat=float(res[i][1])
                  lon=float(res[i][2])
                  #calcula a distancia de dois pontos

      v=vinc_dist(f,a,math.radians(lat_min),math.radians(lon_min),math.ra
dians(lat),math.radians(lon))
                  dist=v[0]
                  angle = (math.pi/2) - v[1]
                  x=dist*math.cos(angle)
                  y= dist*math.sin(angle)
                  #nodes_temp_x='$node_(' + str(i) + ') set X_ ' + str(x)
+ "\n"
                  #nodes_temp_y='$node_(' + str(i) + ') set Y_ ' + str(y)
+ "\n"
                  #nodes_temp_z='$node_(' + str(i) + ') set Z_ 0.0\n'
```

```
                  nodes_temp_x_y_z = 'vector(' + str(x) + ',' + str(y) +
',0.0)';
                  bssid_ = str(keys[i])
                  #number = "node: " + str(i) + "\n";
                  #nodes = nodes + bssid_ + nodes_temp_x + nodes_temp_y +
nodes_temp_z
                  #nodes = nodes + number + bssid_ + nodes_temp_x_y_z
                  #nodes = nodes + nodes_temp_x_y_z + " * " + bssid_ + "
|\n"
                  nodes = nodes + str(x) + "*" + str(y) + "\n"
                  i+=1

            print "A gerar o ficheiro " + file_output_name + " ... \n"
            ns3_file(nodes, file_output_name)
      except IOError:
            print "Erro a gerar o ficheiro!"
            sys.exit(1)
else:
      Usage()
      sys.exit(1)
```

IV – *Script* gps2BG.py

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009

OBJECTIVO: Processar a informacao obtida sobre os pontos localizados

SINTAXE:    script.py [ficheiro entrada] [alcance (m)] [ficheiros saida
(opcional)]
      [ficheiro entrada]      Apenas converte ficheiros *.gpsxml


SAIDA: [ficheiros saida: .edge & .weight & .coor & .nedge]
      Por omissao: [ficheiro_entrada].{edge|weight|coor|n}
"""

#Funcao que calcula a distancia em metros entre dois pontos
#com coordenadas usando a formula inversa de Vincenty para elipsoides
def vinc_dist(  f,  a,  phi1,  lembda1,  phi2,  lembda2 ) :

        if (abs( phi2 - phi1 ) < 1e-8) and ( abs( lembda2 - lembda1) <
1e-8 ) :
          return 0.0, 0.0, 0.0

        two_pi = 2.0*math.pi

        b = a * (1.0 - f)

        TanU1 = (1-f) * math.tan( phi1 )
        TanU2 = (1-f) * math.tan( phi2 )

        U1 = math.atan(TanU1)
        U2 = math.atan(TanU2)

        lembda = lembda2 - lembda1
        last_lembda = -4000000.0                # an impossibe value
        omega = lembda

        # Iterate the following equations,
        #  until there is no significant change in lembda

        while ( last_lembda < -3000000.0 or lembda != 0 and abs(
(last_lembda - lembda)/lembda) > 1.0e-9 ) :

          sqr_sin_sigma = pow( math.cos(U2) * math.sin(lembda), 2) + \
                pow( (math.cos(U1) * math.sin(U2) - \
                math.sin(U1) *  math.cos(U2) * math.cos(lembda) ), 2 )

          Sin_sigma = math.sqrt( sqr_sin_sigma )

          Cos_sigma = math.sin(U1) * math.sin(U2) + math.cos(U1) *
math.cos(U2) * math.cos(lembda)
```

```
          sigma = math.atan2( Sin_sigma, Cos_sigma )

          Sin_alpha = math.cos(U1) * math.cos(U2) * math.sin(lembda) /
math.sin(sigma)
          alpha = math.asin( Sin_alpha )

          Cos2sigma_m = math.cos(sigma) - (2 * math.sin(U1) *
math.sin(U2) / pow(math.cos(alpha), 2) )

          C = (f/16) * pow(math.cos(alpha), 2) * (4 + f * (4 - 3 *
pow(math.cos(alpha), 2)))

          last_lembda = lembda

          lembda = omega + (1-C) * f * math.sin(alpha) * (sigma + C *
math.sin(sigma) * \
                (Cos2sigma_m + C * math.cos(sigma) * (-1 + 2 *
pow(Cos2sigma_m, 2) )))


        u2 = pow(math.cos(alpha),2) * (a*a-b*b) / (b*b)

        A = 1 + (u2/16384) * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))

        B = (u2/1024) * (256 + u2 * (-128+ u2 * (74 - 47 * u2)))

        delta_sigma = B * Sin_sigma * (Cos2sigma_m + (B/4) * \
                (Cos_sigma * (-1 + 2 * pow(Cos2sigma_m, 2) ) - \
                (B/6) * Cos2sigma_m * (-3 + 4 * sqr_sin_sigma) * \
                (-3 + 4 * pow(Cos2sigma_m,2 ) )))

        s = b * A * (sigma - delta_sigma)

        alpha12 = math.atan2( (math.cos(U2) * math.sin(lembda)), \
                (math.cos(U1) * math.sin(U2) - math.sin(U1) *
math.cos(U2) * math.cos(lembda)))

        alpha21 = math.atan2( (math.cos(U1) * math.sin(lembda)), \
                (-math.sin(U1) * math.cos(U2) + math.cos(U1) *
math.sin(U2) * math.cos(lembda)))

        if ( alpha12 < 0.0 ) :
                alpha12 =  alpha12 + two_pi
        if ( alpha12 > two_pi ) :
                alpha12 = alpha12 - two_pi

        alpha21 = alpha21 + two_pi / 2.0
        if ( alpha21 < 0.0 ) :
                alpha21 = alpha21 + two_pi
        if ( alpha21 > two_pi ) :
                alpha21 = alpha21 - two_pi

        return s, alpha12,  alpha21

#Funcao que extrai toda a informacao do ficheiro *.gps
#le a i linha do ficheiro *.gps em formato xml
#Input:linha, ficheiro
#Output:end (sinal de controlo), bssid, source, lat, lon, signal
```

```
def xml_parser(i,file_input):
      try:
            gps_run = xml_file.getroot()
            #Output
            end = 0
            bssid = gps_run[i].get("bssid")
            source = gps_run[i].get("source")
            lat = gps_run[i].get("lat")
            lon = gps_run[i].get("lon")
            signal = gps_run[i].get("signal_dbm")
      except Exception, inst:
            #Output
            end = 1
            bssid = -999
            source = -999
            lat = 999
            lon = 999
            signal = -999
      return      end, bssid, source, lat, lon, signal

#Funcao que devolve o ponto com maior latitude e menor longitude
#Input: ficheiro
#Output: lat_min, lon_min
def corner_values(file_input_name):
      lat_min = 999
      lon_min = 999
      i=1
      while(1):
            res = xml_parser(i,file_input_name)
            if res[0] ==1:
                  break
            lat = float(res[3])
            lon = float(res[4])
            #exclui linhas do gps e linhas dos clientes ligados aos aps
            if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG" and
res[1] != "00:00:00:00:00:00"):
                  #procura pelo ponto de menor latitude e menor longitude
                  #NOTA: esta regra aplica-se a locais com latitudes
decimais positivas
                  # e longitudes decimais negativas, neste caso
específico: Aveiro
                  if lat < lat_min:
                        lat_min=lat
                  if lon < lon_min:
                        lon_min=lon
            i+=1
      return lat_min, lon_min

#Funcao que gera duas listas para filtrar a informacao extraida do
ficheiro *.gps
#Input: ficheiro
#Output: keys (lista com os macs dos aps), info (lista com todos os ponto
capturados dos aps)
def create_lists(file_input_name):
      keys = []
      info = []
      i=1
      j=0
      while(1):
```

```
                  res = xml_parser(i,file_input_name)
                  if res[0] ==1:
                        break
                  bssid = str(res[1])
                  #exclui linhas do gps e linhas dos clientes ligados aos
aps
                  if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG"
and res[1] != "00:00:00:00:00:00"):
                        lat = res[3]
                        lon = res[4]
                        signal = res[5]
                        if bssid in keys:
                              index=keys.index(bssid)
                              info[index].append([signal,lat,lon])
                        else:
                              keys.append(bssid)
                              index=keys.index(bssid)
                              info.insert(index,[[signal,lat,lon]])

                  i+=1
      return keys, info


#Funcao que gera o ficheiro os saida
#Input: string edges, string weights,  nome do ficheiro edge, nome do
ficheiro weight
#Output: ficheiro edge, ficheiro weight
def bgraph_file(edges, weights, coor, nedge, n_nodes,
file_output_name_edge,
file_output_name_weight,file_output_name_coor,file_output_name_n):
      try:
            file_output_edge = open(file_output_name_edge,'w')
            file_output_edge.writelines (edges)
            file_output_edge.close()

            file_output_weight = open(file_output_name_weight,'w')
            file_output_weight.writelines (weights)
            file_output_weight.close()

            file_output_coor = open(file_output_name_coor,'w')
            file_output_coor.writelines (coor)
            file_output_coor.close()

            file_output_n = open(file_output_name_n,'w')
            file_output_n.writelines (n_nodes)
            file_output_n.writelines ("\n")
            file_output_n.writelines (nedge)
            file_output_n.close()
            print "Ficheiros gerados!\n"
      except IOError:
            print "ERRO ao criar os ficheiros de saida!\n"
            sys.exit(1)

#Funcao que devolve para cada ap a latitude e a longitude da amostra com
maior sinal
#Input: info (lista)
#Output: res (lista)
def best_signal(info):
      res = []
```

```
	for i in range(len(info)):
		res.append(info[i][0])
		for j in range(len(info[i])):
			if int(info[i][j][0]) < int(res[i][0]):
				res[i] = info[i][j]
	return res

args = sys.argv[1:]
if args:
	file_input_name=args[0]
	try:
		alcance = int(args[1])
		print "\nAlcance considerado: " + str(alcance) + " m\n"
	except Exception, e:
		print "\nERRO!!! >> Insira um numero para o alcance!\n"
		sys.exit()

	if (len(args)==3):
		file_output_name_edge=args[2]+".edge"
		file_output_name_weight=args[2]+".weight"
		file_output_name_coor=args[2]+".coor"
		file_output_name_n=args[2]+".n"
	else:
		file_output_name_edge=file_input_name+".edge"
		file_output_name_weight=file_input_name+".weight"
		file_output_name_coor=file_input_name+".coor"
		file_output_name_n=file_input_name+".n"
	print "A ler o ficheiro " + file_input_name + "\n"
	try:
		xml_file = ET.parse(file_input_name)
		print "\n   Ficheiro lido: %s!\n" %file_input_name
	except IOError:
		print "\n   Ficheiro '%s' NAO existe!\n" %file_input_name
		sys.exit(1)
	try:
		edges = ""
		weights = ""
		edges_temp = ""
		weights_temp = ""
		num_edges = 0
		coor = ""
		nedge= ""
		print "A extrair informacao do ficheiro...\n"
		#extrai a informacao do gps e cria duas listas
		l=create_lists(file_input_name)
		print "     Ok!\n"
		#retorno da funcao create_lists
		keys = l[0]
		info = l[1]
		n_nodes = str(len(info))
		print "Numero total de nos encontrados: " + n_nodes + "\n"

		#procura nos dados extraidos, para cada ap as coordenadas
		#     correspondentes ao maior sinal
		print "A processar informacao...\n"
		res = best_signal(info)
		print "     Ok!\n"

		#valores da elipsoide WGS-84
```

```
            a=6378137
            f=1/298.257223563

            i=0
            j=0
            print "A calcular distancias entre os pontos ...\n"
            for i in range(len(res)):

                  lat_i=float(res[i][1])
                  lon_i=float(res[i][2])

                  coor = coor + str(lat_i) + " " + str(lon_i) + "\n"

                  for j in (range(len(res))):
                        if i == j:
                              continue
                        #print "Node " + str(i) + " -> " + str(j) + "\n"
                        edge = str(num_edges)+" "+str(i) + "*" + str(j)
                        rev_edge = " "+str(j) + "*" + str(i) + "\n"

                        if edges.find(rev_edge)  != -1:
                              #print "Ignoring " + edge + "\n"
                              continue

                        lat_j=float(res[j][1])
                        lon_j=float(res[j][2])
                        #calcula a distancia de dois pontos


      v=vinc_dist(f,a,math.radians(lat_i),math.radians(lon_i),math.radian
s(lat_j),math.radians(lon_j))
                        dist = round(float(v[0]),4)
                        if dist <= alcance:
                              edges = edges +  edge + "\n"
                              weights = weights + str(num_edges) + " "+
str(dist) + "\n"
                              num_edges+=1
                        j+=1
                  i+=1
            nedge = str(num_edges)
            print "     Ok!\n"
            print "A gerar os ficheiros:\n\t" + file_output_name_edge +
"\n\t" + file_output_name_weight + "\n\t" + file_output_name_coor +
"\n\t" + file_output_name_n + "\n\t" + " ... \n"
            bgraph_file(edges, weights, coor, nedge,
n_nodes,file_output_name_edge, file_output_name_weight,
file_output_name_coor, file_output_name_n)
            print "\nFIM\n"

      except IOError:
            print "Erro a gerar os ficheiros!"
            sys.exit(1)
else:
      Usage()
      sys.exit(1)
```

V – Código para o simulador NS-2

Ficheiro Simul.tcl:

```
#
#     Filipe Manuel Dias Ferreira | MIECT 2009
#     Simulado NS-2 : *.tcl
#
# Define opcoes
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy            ;# network interface
type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue
type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             5480                       ;# number of
mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)              5500                       ;# X dimension of
topography
set val(y)              5500                       ;# Y dimension of
topography
set val(stop)           600169920                  ;# time of simulation
end 120+(10+10)*5478*5478+120

set ns_             [new Simulator]

Phy/WirelessPhy set freq_ 2.472e9
Phy/WirelessPhy set bandwidth_ 54Mb
Phy/WirelessPhy set Pt_ 0.033

Mac/802_11 set dataRate_ 54Mb
Mac/802_11 set basicRate_ 2Mb

remove-all-packet-headers
add-packet-header AODV ARP LL MAC CBR IP TCP ACK FTP

set tracefd       [open sim-out.tr w]
#set namtrace      [open sim-out.nam w]
set topology       nodes.ns2
set traffic   simul.trf

$ns_ trace-all $tracefd
#$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
#$ns_ use-newtrace
$ns_ use-scheduler Heap

# Cria a topologia
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
```

```
# Configura os nos

$ns_ node-config -adhocRouting $val(rp) \
                  -llType $val(ll) \
                  -macType $val(mac) \
                  -ifqType $val(ifq) \
                  -ifqLen $val(ifqlen) \
                  -antType $val(ant) \
                  -propType $val(prop) \
                  -phyType $val(netif) \
                  -channelType $val(chan) \
                  -topoInstance $topo \
                  -agentTrace ON \
                  -routerTrace OFF \
                  -macTrace OFF \
                  -movementTrace OFF

puts "Creating nodes"
for {set i 0} {$i < $val(nn) } { incr i } {
      set node_($i) [$ns_ node]
}

puts "Loading topology"

# Define a topologia
source $topology

puts "Loading traffic pattern"

# Define trafego
source $traffic

# Define a posicao inicial dos nos
puts "Setting node initial position for NAM"
for {set i 0} {$i < $val(nn)} { incr i } {
      $ns_ initial_node_pos $node_($i) 5
}



# Indica quando termina a simulacao aos nos

for {set i 0} {$i < $val(nn) } { incr i } {
    $ns_ at $val(stop) "$node_($i) reset";
}

# Fim da simulacao

$ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
$ns_ at $val(stop) "stop"
$ns_ at $val(stop) "puts \"end simulation\" ; $ns_ halt"


proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
 #   close $namtrace
```

```
}

$ns_ run
```

Ficheiro Simul.trf:

```
#
#     Filipe Manuel Dias Ferreira | MIECT 2009
#     Simulador NS-2 : *.trf
#

# Associa um agente gerador d trafego
# a cada no

for {set k 0} {$k < $val(nn)} {incr k} {
      set tcp_($k) [new Agent/TCP/Newreno]
      $tcp_($k) set class_ 2
      $ns_ attach-agent $node_($k) $tcp_($k)

      set ftp_($k) [new Application/FTP]
      $ftp_($k) attach-agent $tcp_($k)

      set sink_($k) [new Agent/TCPSink]
      $ns_ attach-agent $node_($k) $sink_($k)
}

# Define o tempo envio

set start 10
set duration 5
set interval 5
set stop [ expr $start + $duration]

# Envia de todos para todos os nos pacotes UDP (ftp)

for {set j 0} {$j < $val(nn)} {incr j} {
      for {set i 0} {$i < $val(nn)} {incr i} {
            $ns_ connect $tcp_($j) $sink_($i)
            $ns_ at $start "$ftp_($j) start"
            $ns_ at $stop "$ftp_($j) stop"

            set start [expr [ expr $start + $stop] + $interval]
            set stop [expr $start + $duration]
      }
}
```

VI – Código para o simulador NS-3

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Filipe Manuel Dias Ferreira | MIECT 2009
 * Simulador NS-3
*/


#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/mobility-module.h"
#include "ns3/mobility-helper.h"
#include "ns3/contrib-module.h"
#include "ns3/position-allocator.h"
#include "ns3/vector.h"
#include "ns3/wifi-module.h"
#include "ns3/common-module.h"

#include <iostream>
#include <fstream>
#include <string.h>
#include <sstream>

using namespace std;
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Simul");


int main (int argc, char *argv[])
{
            LogComponentEnable("OnOffApplication",LOG_LEVEL_INFO);


// Tempo Total  120+(10+10)*5478*5478+120 (seg)
        uint32_t stopTime = 600169920;

// Numero de nos
        uint32_t nr_nodes = 5478;

// Porta UDP para a aplicacao
        uint16_t port = 9;

// Tempos da aplicacao
        uint32_t startTimeApp = 10;
        uint32_t durationTime = 5;
        uint32_t intervalTime = 5;
        uint32_t stopTimeApp = startTimeApp + durationTime + intervalTime
;


        Config::SetDefault ("ns3::OnOffApplication::PacketSize",
StringValue ("64"));
        Config::SetDefault ("ns3::OnOffApplication::DataRate",
StringValue ("256b/s"));
```

```
        NodeContainer nodes;
        nodes.Create (nr_nodes+1);

// Cada no tem um canal associado e um netdevice
// neste caso: WifiChannel, WifiNetDevice
        WifiHelper wifi;
        NqosWifiMacHelper mac = NqosWifiMacHelper::Default();
        mac.SetType("ns3::AdhocWifiMac");

        wifi.SetRemoteStationManager
("ns3::ConstantRateWifiManager","DataMode",StringValue ("wifia-54mbs"));

        YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
        YansWifiChannelHelper wifiChannel =
YansWifiChannelHelper::Default ();
        wifiPhy.SetChannel (wifiChannel.Create ());

        NetDeviceContainer devices = wifi.Install (wifiPhy, mac,nodes);

// Protocolo

        NS_LOG_INFO ("Enabling OLSR routing on all backbone nodes");

        OlsrHelper olsr;

// Adiciona o protocolo IPv4 aos nos do container
        InternetStackHelper internet;
        internet.SetRoutingHelper (olsr);
        internet.Install (nodes);

        internet = InternetStackHelper ();

// Distribui IPs pela rede
        Ipv4AddressHelper ipAddrs;
        ipAddrs.SetBase ("192.168.0.0", "255.255.224.0");
        ipAddrs.Assign (devices);

// Cria a topologia
        Ptr<ListPositionAllocator> list_nodes =
CreateObject<ListPositionAllocator> ();

        MobilityHelper mobility;

        NS_LOG_INFO ("Creating Topology");

// Le a topologia do ficheiro *.ns3
        string line;
        string x_="0.0",y_="0.0";
        double x=0.0,y=0.0;

        ifstream file ("/home/fdias/files/old/nodes.ns3");

        if (file.is_open()) {
                NS_LOG_INFO ("Reading nodes.ns3");
                while (! file.eof()) {
                        if (getline(file,line,'*')) {
                                x_= line.c_str();
                                if (getline(file,line,'\n')) {
```

```
                                        y_=line.c_str();
                                }
                        }
                        x=atof(x_.c_str());
                        y=atof(y_.c_str());
                        Vector vector_i(x,y,0.0);
                        list_nodes->Add(vector_i);
                }
                file.close();
                NS_LOG_INFO ("End file read");
        } else NS_LOG_INFO ("Read file ERROR!");

        mobility.SetPositionAllocator (list_nodes);
        mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
        mobility.Install (nodes);

// Configura as aplicacoes

        NS_LOG_INFO ("Creating Aplications");

// Aplicacao UDP

        uint16_t i;
        uint16_t j;
        std::stringstream ss;

        ss << "Creating Applications in  " << nr_nodes << " nodes" <<
std::endl;

        NS_LOG_INFO (ss.str());

        for (i=0;i<nr_nodes;i++) {
                Ptr<Node> appSource = NodeList::GetNode (i);
                ss << "Start Node: " << appSource->GetId() << std::endl;
                        NS_LOG_INFO(ss.str());
                for (j=0;j<nr_nodes;j++) {
                                //if (i!=j){continue;}
                        Ptr<Node> appSink = NodeList::GetNode (j);
                        ss << "End Node: " << appSink->GetId()<<
std::endl;
                                NS_LOG_INFO(ss.str());

                        Ipv4Address remoteAddr = appSink->GetObject<Ipv4>
()->GetAddress(1, 0).GetLocal ();

                        OnOffHelper onoff
("ns3::UdpSocketFactory",Address (InetSocketAddress (remoteAddr, port)));
                        onoff.SetAttribute ("OnTime", RandomVariableValue
(ConstantVariable (1)));
                        onoff.SetAttribute ("OffTime",
RandomVariableValue (ConstantVariable (0)));

                        ApplicationContainer apps = onoff.Install
(appSource);
                        apps.Start (Seconds (startTimeApp));
                        apps.Stop (Seconds (stopTimeApp));
```

```
                        PacketSinkHelper sink
("ns3::UdpSocketFactory",InetSocketAddress (Ipv4Address::GetAny (),
port));
                        apps = sink.Install (appSink);
                        apps.Start (Seconds (startTimeApp));

                        startTimeApp += durationTime + intervalTime;
                        stopTimeApp = startTimeApp + durationTime;

                }
        }

        Simulator::Stop(Seconds (stopTime));

// Processa o trace da simulacao

        NS_LOG_INFO ("Configuring Tracing");
        std::ofstream ascii;
        ascii.open ("simul-2.tr");
        NS_LOG_INFO ("Enabling Ascii file");
        InternetStackHelper::EnableAscii (ascii,nodes);

// Corre a simulacao

        NS_LOG_INFO ("Running Simulation!");
        Simulator::Run ();
        Simulator::Destroy ();
        return 0;

}
```

VII – Simulador (simul.cpp)

```
//============================================================================
//    Filipe Manuel Dias Ferreira | MIECT 2009
//    Simulador com livraria BoostGraph
//
//============================================================================
#include <boost/config.hpp>
#include <iostream>
#include <fstream>

#include <boost/graph/graph_traits.hpp>
#include <boost/graph/adjacency_list.hpp>
#include <boost/graph/dijkstra_shortest_paths.hpp>
#include <boost/graph/bellman_ford_shortest_paths.hpp>
#include <boost/graph/graphviz.hpp>
#include <boost/property_map/property_map.hpp>

using namespace boost;
using namespace std;

typedef struct{
      int hops;
      int cost;
      int id;
}node_neighbor_t;

typedef struct{
      int id;
      double latitude;
      double longitude;
      vector<node_neighbor_t*> neighbors;
}node_t;

typedef adjacency_list < listS, vecS, undirectedS,no_property, property <
edge_weight_t, float > > graph_t;
typedef graph_traits < graph_t >::vertex_descriptor vertex_descriptor;
typedef graph_traits < graph_t >::edge_descriptor edge_descriptor;
typedef pair<int, int> Edge;


int getNumberHops(int x,int y, vector< vertex_descriptor > *p)
{
      int pred = p[x][y];
      if (pred == x){
            return 1;
      }
      else return getNumberHops(x,pred,p)+1;
}

void usage()
{
      cout << endl << "Filipe Dias | MIECT @2009" <<endl;
      cout << "SINTAXE:\n\tsimul
[directorio/ficheiro].{n|edge|weights|coor} [directorio para ficheiros
saida] [nome ficheiros de saida]" <<endl<<endl;
      cout << "NOTA: \n\tapenas introduzir o nome de ficheiro de entrada
e saida sem extensao!" <<endl<< endl;
```

```
      cout << "FICHEIROS SAIDA: \n\t[nome ficheiro
entrada].{dot|links|plot|results}" <<endl<<endl;
}

int main(int argc, char *argv[])
{
  if (argc <4) {usage();exit(-1);}
  else{

  string file_input = argv[1];
  string fn = file_input + ".n";
  string fedge = file_input + ".edge";
  string fweights = file_input + ".weight";
  string fcoor = file_input + ".coor";

  string dir_output = argv[2];
  string file_output = argv[3];
  string fdot = dir_output + file_output + ".dot";
  string flinks = dir_output + file_output + ".links";
  string fto_plot = dir_output + file_output + ".plot";
  string fresults = dir_output + file_output + ".results";

  cout << endl << "INICIO da simulacao" << endl << endl ;

  vector< node_t* > node_list;

  cout << "A carregar numero de nos e de arestas " << endl;
  ifstream file_n (fn.c_str());
  string l = "";
  int aux_l[2];
  int li=0;
  if (file_n.is_open()){
        cout << " Ficheiro " << fn.c_str()<<" ..." << endl;
        while (!file_n.eof()){
            getline(file_n,l);
            aux_l[li]=atoi(l.c_str());
            li++;
        }
        file_n.close();
        cout << "     Fim da leitura com sucesso!" << endl << endl;
  }
      else{
            cout << "     ERRO na leitura do ficheiro!" << endl << endl;
            exit(-1);}

  const int num_nodes = aux_l[0];
  const int num_edges = aux_l[1];

  cout << "Numero de nos: " << num_nodes << endl << endl;
  cout << "Numero de arestas: " << num_edges << endl << endl;

  cout << "A criar array vazio de arestas..." << endl;
  Edge edge_array[num_edges];
  memset(edge_array,0,sizeof(Edge)*num_edges);
  cout << "Ok!" << endl;
  cout << " A ler arestas do ficheiro..." << endl;
  ifstream file_edge (fedge.c_str());
  string line_edge;
  unsigned i_edge = 0, j_edge = 0,n_edge= 0;
```

```
  if (file_edge.is_open()){
        cout << "       Ficheiro "<< fedge.c_str() <<" ..." << endl;
        while (!file_edge.eof()){
            if(getline(file_edge,line_edge)){
                  if(sscanf(line_edge.c_str(),"%u
%u*%u\n",&n_edge,&i_edge,&j_edge) != 3)
                  {
                        cout << "Erro ao interpretar a linha: " <<
line_edge << endl;
                        exit(-1);
                  }
                  edge_array[n_edge]=Edge(i_edge,j_edge);
            }
        }
        file_edge.close();
        cout << "       Fim da leitura com sucesso!" << endl << endl;
  }
  else{
        cout << "       ERRO na leitura do ficheiro!" << endl << endl;
        exit(-1);
  }

  cout << "A criar um array vazio de pesos..." << endl;
  float weights[num_edges];
  memset(weights,0,num_edges*sizeof(float));
  cout << " Ok!" << endl;

  cout << " A ler pesos do ficheiro..." << endl;
  ifstream file_weight (fweights.c_str());
  string line_weight;
  float w=0;
  int lines_weight=0;
  if (file_weight.is_open()){
        cout << "       Ficheiro "<< fweights.c_str() << "..." << endl;
        while (!file_weight.eof()){
              if(getline(file_weight,line_weight)){
                  if(sscanf(line_weight.c_str(),"%d %f",&lines_weight,&w)
!= 2)
                  {
                        cout << "Erro ao interpretar a linha: " <<
line_weight << endl;
                        exit(-1);
                  }
                 weights[lines_weight]=w;
              }
        }
       file_weight.close();
       cout << "       Fim da leitura com sucesso!" << endl << endl;
  }
  else{
        cout << "       ERRO na leitura do ficheiro!" << endl << endl;
        exit(-1);
  }


  cout << "A ler coordenadas dos pontos..." << endl;
  fstream file_coor (fcoor.c_str());
  string line_coor;
```

```
  float lat[num_nodes];
  float lon[num_nodes];

  memset(lat,0,sizeof(float)*num_nodes);
  memset(lon,0,sizeof(float)*num_nodes);

  int aux=0;
  if (file_coor.is_open()){
        cout << " Ficheiro " << fcoor.c_str() << "..." << endl;
        while (!file_coor.eof()){
              if(getline(file_coor,line_coor)){
                    if(sscanf(line_coor.c_str(),"%f
%f\n",&lat[aux],&lon[aux]) != 2){
                          cout << "Erro a interpretar a linha: " <<
line_coor << endl;
                          exit(-1);}
              aux++;
              }

        }
        file_coor.close();
        cout << "     Fim da leitura com sucesso!" << endl << endl;
  }
  else{
      cout << "     ERRO na leitura do ficheiro!" << endl << endl;
      exit(-1);}
  cout << " Ok!" << endl;

  cout << "A criar grafo..." << endl;
  graph_t g(num_nodes);

  cout << "       A inserir arestas..." << endl;
   property_map<graph_t, edge_weight_t>::type weightmap =
get(edge_weight, g);
   for (size_t j = 0; j < num_edges; ++j) {
    edge_descriptor e; bool inserted;
    tie(e, inserted) = add_edge(edge_array[j].first,
edge_array[j].second, g);
    weightmap[e] = weights[j];
   }
   cout << "      Ok!" << endl;

  cout << "       A numerar os vertices..." << endl << endl;
  vertex_descriptor s[num_nodes];
  memset(s,0,sizeof(vertex_descriptor)*num_nodes);
  for (int k = 0; k < num_nodes; k++){
        s[k]=vertex(k, g);
  }
  cout << " Ok!" << endl;


  property_map<graph_t, vertex_index_t>::type indexmap =
get(vertex_index, g);

  cout << "A Aplicar o algoritmo dijkstra ao grafo..." << endl;

  vector<vertex_descriptor> *p = new
vector<vertex_descriptor>[num_nodes];
  vector<int> *d = new vector<int>[num_nodes];
```

```
  for (int i = 0; i < num_nodes; i++)
  {
            p[i].reserve(num_nodes);
            d[i].reserve(num_nodes);
            node_t node;
            node.id = i+1;
            node.latitude = lat[i];
            node.longitude = lon[i];
            node_list.push_back(&node);
  }

  cout << "Vertice: " << endl;
  for (int k=0; k < num_nodes;k++){
            dijkstra_shortest_paths(g, s[k], &p[k][0], &d[k][0],
weightmap, indexmap,less<int>(),
closed_plus<int>(),(numeric_limits<int>::max)(),
0,default_dijkstra_visitor());
            cout << "\r" << k;
  }
  cout << " Ok!" << endl;

  ofstream links (flinks.c_str());
  ofstream fdt (fdot.c_str());
  cout << "A escrever resultados para o ficheiro: " << flinks.c_str() <<
" ..." << endl;
  cout << "A escrever o ficheiro dot: " << fdot.c_str() << "..." << endl;

  fdt << "digraph A {\n"
        << "  rankdir=LR\n"
        << "  size=\"5,3\"\n"
        << "  ratio=\"fill\"\n"
        << "  edge[style=\"bold\"]\n" << "node[shape=\"circle\"]\n";

  links << "Distances and parents:" << std::endl;
  int hops[num_nodes][4];
  memset(hops,0,sizeof(int)*num_nodes*4);
  int total_hops[4];
  memset(total_hops,0,sizeof(int)*4);

  for (int y=0; y < num_nodes; y++){
       for (int z=0; z<num_nodes; z++){

             links << "vertex " << y << " -> "<< z << ": distance = " <<
d[y][z] << ", parent = " << p[y][z] << "\n";
             if ((y!=z) && (d[y][z]!=2147483647))
             {
                 fdt << y << " -> " << z << "\n";
                 node_t *node = node_list[y];
                 node_neighbor_t *neighbor = new node_neighbor_t();
                 neighbor->id = z;
                 neighbor->hops = getNumberHops(y,z,p);
                 if (neighbor->hops <= 10){
                       hops[y][2]++;
                       total_hops[2]++;
                       if(neighbor->hops <= 5){
                             hops[y][1]++;
                             total_hops[1]++;
                             if(neighbor->hops == 1){
                                   hops[y][0]++;
```

```
                                    total_hops[0]++;
                                }
                            }
                        }
                        neighbor->cost = d[y][z];
                        node->neighbors.push_back(neighbor);
                        hops[y][3]++;
                        total_hops[3]++;

                    }
                }
            }

    fdt << "}";
    cout << "       Ok!" << endl;

    cout << "A criar ficheiro plot : " << fto_plot.c_str() << " ..." <<
endl;
    ofstream to_plot (fto_plot.c_str());
    to_plot << "node latitude longitude 1Hop 5Hops 10Hops multiplesHops" <<
endl;

    float m_mult_hops,p_mult_hops;m_mult_hops = p_mult_hops = 0;
    float m_1_hops,p_1_hops;m_1_hops = p_1_hops = 0;
    float m_5_hops, p_5_hops;m_5_hops = p_5_hops = 0;
    float m_10_hops, p_10_hops;m_10_hops = p_10_hops = 0;

    int tmp, tmp_1, tmp_5, tmp_10;
    tmp = tmp_1 = tmp_5 = tmp_10 = 0;

    for (int a=0; a < num_nodes ; a++){
        to_plot << a << " " << lat[a] << " " << lon[a] << " " << hops[a][0]
<< " " << hops[a][1] << " " << hops[a][2] << " " << hops[a][3] << endl;
    }

    cout << " Ok!" << endl;

    cout << "A escrever resultados estatiscos para o ficheiro: " <<
fresults.c_str() << " ..." <<endl;
    ofstream res (fresults.c_str());

    m_mult_hops = (total_hops[3]/num_nodes);
    m_1_hops = (total_hops[0]/num_nodes);
    m_5_hops = (total_hops[1]/num_nodes);
    m_10_hops = (total_hops[2]/num_nodes);

    p_mult_hops = (m_mult_hops/num_nodes) * 100.00;
    p_1_hops = (m_1_hops/num_nodes) * 100.00;
    p_5_hops = (m_5_hops/num_nodes) * 100.00;
    p_10_hops = (m_10_hops/num_nodes) * 100.00;

    res << "RESULTADOS" << endl << endl;
    res << "Conectividade a 1 hop(media):" << endl << endl;
    res << "   " << p_1_hops << " %" << endl;
    res << "   " << (int)m_1_hops << " nos/vertices em " << num_nodes << "
nos/vertices" << endl << endl;
    res << "Conectividade a 5 hops(media):" << endl << endl;
    res << "   " <<  p_5_hops << " %" << endl;
```

```
  res << "  " << (int)m_5_hops << " nos/vertices em " << num_nodes << "
nos/vertices" << endl << endl;
  res << "Conectividade a 10 hops(media):" << endl << endl;
  res << "  " <<  p_10_hops << " %" << endl;
  res << "  " << (int)m_10_hops << " nos/vertices em " << num_nodes << "
nos/vertices" << endl << endl;
  res << "Conectividade a multiplos hops(media):" << endl << endl;
  res << "  " <<  p_mult_hops << " %" << endl;
  res << "  " << (int)m_mult_hops << " nos/vertices em " << num_nodes <<
" nos/vertices" << endl << endl;

  cout  << "    Ok!" << endl;

  cout << endl << "FIM da simulacao!" << endl;

  return EXIT_SUCCESS;
  }
}
```

VIII – Contador estatístico de segurança (count_security.py)

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009

OBJECTIVO: Qualificar a segurança de uma dada zona

SINTAXE:    count_security.py [ficheiro entrada]
      [ficheiro entrada]      Apenas le ficheiros *.netxml
"""

args = sys.argv[1:]
if args:
      file_input=args[0]
      print "\nA ler o ficheiro " + file_input + "\n"
      try:
            read_xml = ET.parse(file_input)
            print "\n   Ficheiro lido: %s!\n" %file_input
      except Exception, e:
            print "\n   Ficheiro '%s' NAO existe!\n" %file_input
            sys.exit()
      try:
            print "\nA processar informacao ...\n"
            n_net = read_xml.getroot().findall("wireless-network")
            no_secu = 0
            secu = 0
            no = 0
            macs = []
            info = []
            for elem in n_net:
                  bssid = elem.findtext("BSSID")
                  if bssid in macs:
                        continue
                  else:
                        macs.append(bssid)
                        index = macs.index(bssid)
                        ssid= elem.find("SSID")
                        enc = ssid.findtext("encryption")
                        info.insert(index,enc)
            for i in range(len(info)):
                  if info[i] == "None":
                        no_secu +=1
                        print "non secure"
                  else:
                        secu +=1
                        print "secure"
            total = no_secu+secu
            print "\nEstatisticas de seguranca: \n"
            print "\tredes seguras: " + str(secu) + "\n"
            print "\tredes inseguras: " + str(no_secu) + "\n"
            print "\tTOTAL : " + str(total) + "\n"
            print "\nFIM!\n"
```

```
        except Exception, e:
                print "\nErro a processar a informacao! \n"
                sys.exit()

else:
        Usage()
        sys.exit(1)
```

IX – Contador estatístico de canais (count_channels.py)

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009

OBJECTIVO: Qualificar o espectro de uma dada zona

SINTAXE:    count_channels.py [ficheiro entrada] [nome ficheiro saida]
      [ficheiro entrada]      Apenas le ficheiros *.netxml


SAIDA: [ficheiro saida]: [nome ficheiro saida].ch
      Por omissao: [ficheiro_entrada].ch
"""

args = sys.argv[1:]
if args:
      file_input=args[0]

      if (len(args)==2):
            file_output=args[1]+".ch"
      else:
            file_output=file_input+".ch"
      print "\nA ler o ficheiro " + file_input + "\n"
      try:
            read_xml = ET.parse(file_input)
            print "\n   Ficheiro lido: %s!\n" %file_input
      except Exception, e:
            print "\n   Ficheiro '%s' NAO existe!\n" %file_input
            sys.exit()
      try:
            print "\nA processar informacao para o ficheiro: %s\n"
%file_output

            w_networks = read_xml.getroot().findall("wireless-network")
            print w_networks
            channels = [0,0,0,0,0,0,0,0,0,0,0,0,0,0]
            macs = []
            info = []
            for elem in w_networks:
                  bssid = elem.findtext("BSSID")
                  if bssid in macs:
                        continue
                  else:
                        macs.append(bssid)
                        index = macs.index(bssid)
                        info.insert(index,elem.findtext("channel"))
            filew = open(file_output, 'w')
                filew.writelines("\nEspectro de canais de " + file_input
+ "\n\n")

            for i in range(len(info)):
```

```
                  channels[int(info[i])]=channels[int(info[i])]+1
            for j in range(len(channels)):
                  filew.writelines(str(j) + ": " + str(channels[j]) +
"\n")
            filew.close()
            print "     OK!\n"
            print "\n FIM \n"
      except Exception, e:
            print "     Erro a processar a informacao! \n"
            sys.exit()

      except Exception, e:
            print "Erro a gerar os ficheiros!"
            sys.exit()
else:
      Usage()
      sys.exit(1)
```

X – Contador estatístico de distâncias (distances.py)

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
import os,sys, getopt,math,re, string
from xml.etree import ElementTree as ET

def Usage():
        print """
Filipe Dias | MIECT @ 2009

OBJECTIVO: Calcular distancia media entre todos os no's localizados

SINTAXE:    script.py [ficheiro entrada]
      [ficheiro entrada]      Apenas le ficheiros *.gpsxml
"""

#Funcao que calcula a distancia em metros entre dois pontos
#com coordenadas usando a formula inversa de Vincenty para elipsoides
def vinc_dist(  f,  a,  phi1,  lembda1,  phi2,  lembda2 ) :

        if (abs( phi2 - phi1 ) < 1e-8) and ( abs( lembda2 - lembda1) <
1e-8 ) :
          return 0.0, 0.0, 0.0

        two_pi = 2.0*math.pi

        b = a * (1.0 - f)

        TanU1 = (1-f) * math.tan( phi1 )
        TanU2 = (1-f) * math.tan( phi2 )

        U1 = math.atan(TanU1)
        U2 = math.atan(TanU2)

        lembda = lembda2 - lembda1
        last_lembda = -4000000.0                # an impossibe value
        omega = lembda

        # Iterate the following equations,
        #  until there is no significant change in lembda

        while ( last_lembda < -3000000.0 or lembda != 0 and abs(
(last_lembda - lembda)/lembda) > 1.0e-9 ) :

          sqr_sin_sigma = pow( math.cos(U2) * math.sin(lembda), 2) + \
                pow( (math.cos(U1) * math.sin(U2) - \
                math.sin(U1) *  math.cos(U2) * math.cos(lembda) ), 2 )

          Sin_sigma = math.sqrt( sqr_sin_sigma )

          Cos_sigma = math.sin(U1) * math.sin(U2) + math.cos(U1) *
math.cos(U2) * math.cos(lembda)

          sigma = math.atan2( Sin_sigma, Cos_sigma )

          Sin_alpha = math.cos(U1) * math.cos(U2) * math.sin(lembda) /
math.sin(sigma)
```

```
          alpha = math.asin( Sin_alpha )

          Cos2sigma_m = math.cos(sigma) - (2 * math.sin(U1) *
math.sin(U2) / pow(math.cos(alpha), 2) )

          C = (f/16) * pow(math.cos(alpha), 2) * (4 + f * (4 - 3 *
pow(math.cos(alpha), 2)))

          last_lembda = lembda

          lembda = omega + (1-C) * f * math.sin(alpha) * (sigma + C *
math.sin(sigma) * \
                (Cos2sigma_m + C * math.cos(sigma) * (-1 + 2 *
pow(Cos2sigma_m, 2) )))


        u2 = pow(math.cos(alpha),2) * (a*a-b*b) / (b*b)

        A = 1 + (u2/16384) * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))

        B = (u2/1024) * (256 + u2 * (-128+ u2 * (74 - 47 * u2)))

        delta_sigma = B * Sin_sigma * (Cos2sigma_m + (B/4) * \
                (Cos_sigma * (-1 + 2 * pow(Cos2sigma_m, 2) ) - \
                (B/6) * Cos2sigma_m * (-3 + 4 * sqr_sin_sigma) * \
                (-3 + 4 * pow(Cos2sigma_m,2 ) )))

        s = b * A * (sigma - delta_sigma)

        alpha12 = math.atan2( (math.cos(U2) * math.sin(lembda)), \
                (math.cos(U1) * math.sin(U2) - math.sin(U1) *
math.cos(U2) * math.cos(lembda)))

        alpha21 = math.atan2( (math.cos(U1) * math.sin(lembda)), \
                (-math.sin(U1) * math.cos(U2) + math.cos(U1) *
math.sin(U2) * math.cos(lembda)))

        if ( alpha12 < 0.0 ) :
                alpha12 =  alpha12 + two_pi
        if ( alpha12 > two_pi ) :
                alpha12 = alpha12 - two_pi

        alpha21 = alpha21 + two_pi / 2.0
        if ( alpha21 < 0.0 ) :
                alpha21 = alpha21 + two_pi
        if ( alpha21 > two_pi ) :
                alpha21 = alpha21 - two_pi

        return s, alpha12,  alpha21

#Funcao que extrai toda a informacao do ficheiro *.gps
#le a i linha do ficheiro *.gps em formato xml
#Input:linha, ficheiro
#Output:end (sinal de controlo), bssid, source, lat, lon, signal
def xml_parser(i,file_input):
      try:
            gps_run = xml_file.getroot()
            #Output
            end = 0
```

```
            bssid = gps_run[i].get("bssid")
            source = gps_run[i].get("source")
            lat = gps_run[i].get("lat")
            lon = gps_run[i].get("lon")
            signal = gps_run[i].get("signal_dbm")
      except Exception, inst:
            #Output
            end = 1
            bssid = -999
            source = -999
            lat = 999
            lon = 999
            signal = -999
      return      end, bssid, source, lat, lon, signal

#Funcao que devolve o ponto com maior latitude e menor longitude
#Input: ficheiro
#Output: lat_min, lon_min
def corner_values(file_input_name):
      lat_min = 999
      lon_min = 999
      i=1
      while(1):
            res = xml_parser(i,file_input_name)
            if res[0] ==1:
                  break
            lat = float(res[3])
            lon = float(res[4])
            #exclui linhas do gps e linhas dos clientes ligados aos aps
            if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG" and
res[1] != "00:00:00:00:00:00"):
                  #procura pelo ponto de menor latitude e menor longitude
                  #NOTA: esta regra aplica-se a locais com latitudes
decimais positivas
                  # e longitudes decimais negativas, neste caso
específico: Aveiro
                  if lat < lat_min:
                        lat_min=lat
                  if lon < lon_min:
                        lon_min=lon
            i+=1
      return lat_min, lon_min

#Funcao que gera duas listas para filtrar a informacao extraida do
ficheiro *.gps
#Input: ficheiro
#Output: keys (lista com os macs dos aps), info (lista com todos os ponto
capturados dos aps)
def create_lists(file_input_name):
      keys = []
      info = []
      i=1
      j=0
      while(1):
                  res = xml_parser(i,file_input_name)
                  if res[0] ==1:
                        break
                  bssid = str(res[1])
```

```
                  #exclui linhas do gps e linhas dos clientes ligados aos
aps
                  if (res[1] == res[2] and res [1] != "GP:SD:TR:AC:KL:OG"
and res[1] != "00:00:00:00:00:00"):
                        lat = res[3]
                        lon = res[4]
                        signal = res[5]
                        if bssid in keys:
                              index=keys.index(bssid)
                              info[index].append([signal,lat,lon])
                        else:
                              keys.append(bssid)
                              index=keys.index(bssid)
                              info.insert(index,[[signal,lat,lon]])

                  i+=1
      return keys, info

#Funcao que devolve para cada ap a latitude e a longitude da amostra com
maior sinal
#Input: info (lista)
#Output: res (lista)
def best_signal(info):
      res = []
      for i in range(len(info)):
            res.append(info[i][0])
            for j in range(len(info[i])):
                  if int(info[i][j][0]) < int(res[i][0]):
                        res[i] = info[i][j]
      return res

args = sys.argv[1:]
if args:
      file_input_name=args[0]
      print "\nA ler o ficheiro " + file_input_name + "\n"
      try:
            xml_file = ET.parse(file_input_name)
            print "\n   Ficheiro lido: %s!\n" %file_input_name
      except Exception, e:
            print "\n   Ficheiro '%s' NAO existe!\n" %file_input_name
            sys.exit()
      try:
            print "A extrair informacao do ficheiro...\n"
            #extrai a informacao do gps e cria duas listas
            l=create_lists(file_input_name)
            print "     Ok!\n"
            #retorno da funcao create_lists
            keys = l[0]
            info = l[1]
            n_nodes = str(len(info))
            print "Numero total de nos encontrados: " + n_nodes + "\n"

            #procura nos dados extraidos, para cada ap as coordenadas
            #     correspondentes ao maior sinal
            print "A processar informacao...\n"
            res = best_signal(info)
            print "     Ok!\n"

            #valores da elipsoide WGS-84
```

```
            a=6378137
            f=1/298.257223563

            #i=0
            #j=0
            dist_t = 0
            print "A calcular distancias entre os pontos ...\n"
            for i in range(len(res)):

                  lat_i=float(res[i][1])
                  lon_i=float(res[i][2])


                  for j in (range(len(res))):
                        if i == j:
                              continue
                        lat_j=float(res[j][1])
                        lon_j=float(res[j][2])
                        #calcula a distancia de dois pontos


      v=vinc_dist(f,a,math.radians(lat_i),math.radians(lon_i),math.radian
s(lat_j),math.radians(lon_j))
                        dist = round(float(v[0]),4)
                        dist_t= dist_t+dist
                        #j+=1
                  #i+=1
            print "Soma distancias: " + str(dist_t) +"\n"
            n = float(n_nodes)
            media = float(dist_t)/float(((n*n)-n))
            print "Distancia media entre os no's: " + str(media) + "\n"

            print "\nFIM\n"

      except Exception, e:
            print "Erro a gerar os ficheiros!"
            sys.exit()
else:
      Usage()
      sys.exit(1)
```